

Probabilistic Modelling of Progressive Filtering

Giuliano Armano

Dept. of Electrical and Electronic Engineering

University of Cagliari

Piazza d'Armi, 09123, Cagliari, Italy

armano@diee.unica.it

The article entitled *Modeling Progressive Filtering*, published on *Fundamenta Informaticae* (Vol. 138, Issue 3, pp. 285-320, July 2015), has been derived from this extended report.

Abstract

Progressive filtering is a simple way to perform hierarchical classification, inspired by the behavior that most humans put into practice while attempting to categorize an item according to an underlying taxonomy. Each node of the taxonomy being associated with a different category, one may visualize the categorization process by looking at the item going downwards through all the nodes that accept it as belonging to the corresponding category. This paper is aimed at modeling the progressive filtering technique from a probabilistic perspective, in a hierarchical text categorization setting. As a result, the designer of a system based on progressive filtering should be facilitated in the task of devising, training, and testing it.

1 Introduction

Classification (or categorization) is a process of labeling data with categories taken from a predefined set, supposed to be semantically relevant to the problem at hand. The absence of an internal structure in the set of categories (or the absence of techniques able to account for this structure) leads to the so-called “flat” models, in which categories are dealt with independently of one another. In the event that the categories are organized in a taxonomy, typically through *is-a* or *part-of* relationships, and assuming that one wants to take into account also this information in order to improve the performance of a categorization system, the corresponding labeling process takes the name of hierarchical classification (*HC*). This research area has received much attention after the explosion of the World Wide Web, in which many problems and the corresponding software applications are based on an underlying taxonomy (e.g., web search with “vertical” search engines, online marketplaces, recommender systems).

This paper is aimed at modeling progressive filtering (hereinafter PF), a hierarchical technique inspired by the behavior that most humans put into practice while attempting to categorize data according to a taxonomy. PF assumes that a top-down categorization process occurs, performed in combination with a set of binary classifiers that mirror the structure of the taxonomy and are entrusted with accepting relevant inputs while rejecting the others. Starting from the root, supposed to be unique, a classifier that accepts an input passes it down to all its offspring (if any), and so on. The typical result consists of activating one or more paths within the taxonomy, i.e., those for which the corresponding classifiers have accepted the given input. While concentrating on hierarchical text categorization (*HTC*) problems, we will be focusing on the following issues: i) Can we predict the expected behavior of a system implemented in accordance with PF when fed with a corpus of documents whose statistical properties are known?, ii) Would it be feasible to separate the statistical information concerning inputs from the intrinsic properties of the classifiers embedded in the given taxonomy?

To my knowledge, no previous work has been done on the above issues, although the reasons for investigating them from a probabilistic perspective are manifold. In particular, a probabilistic model able to estimate the outcomes of a system that implements PF when applied to a real-world task can facilitate taxonomy design, optimization, and assessment. As for *taxonomy design*, the ability to assess in advance an update to the underlying taxonomy could be very useful for a designer. Indeed, adding or removing a node, as well as updating its characteristics, can have a substantial impact on performance, also depending on which metrics the designer wants to maximize. The importance of the model is also motivated by the fact that nowadays many e-businesses (e.g., online stores) resort to human experts to create and maintain the taxonomies considered relevant for their business activities, mainly due to the lack of automatic or semi-automatic tools for taxonomy handling. As for *taxonomy optimization*, let us assume that each classifier in the taxonomy has some parameters for controlling its behavior. The simplest scenario consists of focusing on the acceptance threshold, which typically falls in the range $[0, 1]$. According to this hypothesis, an optimization problem over the threshold space arises, characterized by high time complexity. To make this problem tractable, in our view, one should accept suboptimal solutions while looking at the above issues from a perspective based on three layers (listed from top to bottom): (i) the space of thresholds, (ii) the space of classifiers, and (iii) the space of experiments. For each layer, a source of intractability holds, which can be dealt with by means of approximate models. In particular, a “light” (hence, suboptimal) algorithm for threshold optimization can be used to search the space of thresholds; the mapping between the threshold and the expected behavior of a classifier can be estimated in the space of classifiers,¹ and the probabilistic model introduced in this paper can be used to predict the outcome of a test run on a corpus of documents with known statistical properties. As for *taxonomy assessment*, the possibility of evaluating relevant metrics

¹Many subtle problems arise in the space of classifiers when trying to reflect a change imposed on the space of thresholds. As discussion of these issues is far beyond the scope of this paper, we limit our assertion to the generic recommendations above –intended to overcome the computational issues arising from the need to retrain classifiers.

can provide useful information about the expected behavior of a taxonomy. In particular, checking how the distribution of inputs, together with the characteristics of the embedded classifiers, affect the overall performance of a system compliant with PF can be very important while testing and maintaining a taxonomy.

The rest of the paper is organized as follows: Section 2 briefly recalls some related work, useful for fitting the problem within the current state-of-the-art. Section 3 introduces the concepts deemed most relevant for *HTC*. Section 4 defines PF, first from a probabilistic perspective and then as a linear transformation in the space of (normalized) confusion matrices. Section 5 analyzes how relevant metrics change within a taxonomy. Section 6 provides a critical assessment of PF. Conclusions and future work (Section 7) end the paper.

2 Related Work

In line with the “divide and conquer” philosophy, the main advantage expected from the hierarchical perspective is that the problem is partitioned into smaller subproblems, hopefully easier than the original one, so that each can be effectively and efficiently managed. Beyond this generic consideration, a number of algorithmic and architectural solutions have been experimented. A first rough division can be made between the so-called *local* vs. *global* approach. In the former case an ensemble of classifiers is generated, whereas in the latter a monolithic classifier is generated, able to account for the whole taxonomy. Local approaches seem to interpret the divide and conquer philosophy more properly, as they concentrate on (a typically small) part of the underlying taxonomy while implementing each component of the ensemble. However, the global approach does not prevent local strategies from actually being used to generate a monolithic classifier (e.g., multi-label decision trees).

2.1 Pachinko vs. Probabilistic Machines

In [20], all local approaches that rely on a sequence of top-down decisions take the esoteric name of *pachinko machine*, as they resemble to some extent the corresponding Japanese game. This approach has been widely used with different learning algorithms and techniques: linear classifiers [25], [10], probabilistic classifiers [21], decision rules [18], boosting [15], artificial neural networks (ANNs) [26], support vector machines (SVMs) [29], and in a transductive setting [5]. Moreover, in [20], an extended version of the Pachinko-machine approach is proposed, adding the ability to terminate the categorization process at any intermediate level of the hierarchy.

The so-called *probabilistic machines* adopt an alternative approach, in which all paths are considered simultaneously. Their probabilities are calculated as the product of individual probabilities of categories (for each path), and the leaf categories (i.e., the most probable paths) are selected according to a maximum likelihood criterion. This approach has been used in combination with probabilistic classifiers, [7], with ANNs [14], [35], and with SVMs [2].

It is worth pointing out that Dumais and Chen compared the two local approaches, i.e., Pachinko machine and probabilistic, and found no difference in performance, [13].

2.2 Mapping Between Classifiers and the Underlying Taxonomy

According to the survey paper of [28], a hierarchical approach is better understood when described from two dimensions, i.e., the nature of the given *problem* (or class of problems) and the characteristics of the *algorithm* devised to cope with it (or them). The problem is described by a triple $\langle \Upsilon, \Psi, \Phi \rangle$, where: Υ specifies the type of graph representing the hierarchical classes (i.e., tree or DAG), Ψ indicates whether a data instance is allowed to have class labels associated with a single or multiple paths in the taxonomy, and Φ describes the label depth of the data instances, i.e., full or partial. The algorithm is described by a 4-tuple $\langle \Delta, \Xi, \Omega, \Theta \rangle$, where: Δ indicates whether single or multiple path prediction is performed, Ξ specifies whether leaf-node prediction is mandatory or not, Ω is the taxonomy structure the algorithm can handle (i.e., tree or DAG), and Θ establishes the mapping between classifiers and the underlying taxonomy (i.e., local classifier per node, local classifier per parent node, local classifier per level, and global classifier).

A simple way to categorize the various proposals made in *HC* is to focus on the mapping between classifiers and the underlying taxonomy. Relevant proposals are listed from fine to coarse granularity:

- *Local Classifier per Node.* This approach admits only binary decisions, as each classifier is entrusted with deciding whether the input at hand can be forwarded or not to its children. [10], [13], and [29] are the first proposals in which sequential Boolean decisions are applied in combination with local classifiers per node. In [38], the idea of mirroring the taxonomy structure through binary classifiers is clearly highlighted (the authors call this technique “binarized structured label learning”). In [1], the underlying taxonomy is scattered on the corresponding set of admissible paths which originate from the root (called pipelines). Each component of a pipeline embeds a binary classifier, and pipelines are independently optimized.
- *Local Classifier per Parent Node.* In the seminal work by [21], a document to be classified proceeds top-down along the given taxonomy, each classifier being used to decide to which subtree(s) the document should be sent to, until one or more leaves of the taxonomy are reached. This approach, which requires the implementation of multiclass classifiers for each parent node, gave rise to a variety of actual systems, e.g., [25], [10],[36], and [26].
- *Local Classifier per Level.* This approach can be considered as a boundary between local and global approaches, as the number of outputs per level grows moving down through the taxonomy, soon becoming comparable with the number required for a global classifier. Among the proposals adopting this approach, let us recall [22] and [9].

- *Global Classifier*. One classifier is trained, able to discriminate among all categories. Many global approaches to *HC* have been proposed, e.g., [34], [33], [19], [12], [31], [4], and [20].

According to [29], training systems with a global approach is computationally heavy, as they typically do not exploit different sets of features at different hierarchical levels, and are not flexible, as a classifier must be retrained each time the hierarchical structure changes. On the other hand, although computationally more efficient, local approaches have to make several correct decisions in a row to correctly classify one example, and errors made at top levels are usually not recoverable. Moreover, the categories may lack positive examples at deep levels, making the task of training reliable classifiers difficult.

2.3 Further Relevant Issues for HC

Further relevant issues for *HC* are the way feature selection/reduction is performed and which strategy is adopted to train the classifier(s) embedded in a hierarchical system. Research efforts in this area have focused largely on *HTC*.

Features can be selected according to a global or a local approach (a comparison between the two approaches can be found in [35]). In global approaches, the same set of features is used at any level of the taxonomy, as done with flat categorization. This solution is normally adopted in monolithic systems, where only one classifier is entrusted with distinguishing among all categories in a taxonomy [16, 19]. Variations on this theme can be found in [36] and in [24]. In local approaches, different sets of features are selected for different nodes in the taxonomy, thus taking advantage of dividing a large initial problem into subproblems, e.g., [36]. This is the default choice for Pachinko machines. In a more recent work, [15] suggest that feature selection should pay attention to the topology of the classification scheme. Among other approaches to feature selection, let us recall [23], based on χ -square feature evaluation. As for feature reduction, latent semantic indexing [11] is the most commonly used technique. Based on singular value decomposition [17], it implements the principle that words used in the same contexts tend to have similar meanings.

As for training strategies, according to [6], training sets can be *hierarchical* or *proper*. The former include documents of the subtree rooted in a category as positive examples and documents of the sibling subtrees as negative examples. The latter include documents of a category as positive examples (while disregarding documents from its offspring), and documents of the sibling categories as negative examples. After running several experiments aimed at assessing the pros and cons of the two training strategies, the authors have shown that hierarchical training sets are more effective.

3 Hierarchical Text Categorization

As our work will focus mainly on *HTC*, let us summarize the basic concepts and the issues considered most relevant to this research field (see also [27] and [20]).

3.1 Standard Definitions for HTC

Text Categorization. Text categorization is the task of assigning a Boolean value to each pair $\langle d_j, c_i \rangle \in D \times C$, where D is a domain of documents and $C = \{c_k \mid k = 1, 2, \dots, N\}$ is a set of N predefined categories.

Hierarchical Text Categorization. Hierarchical Text Categorization is a text categorization task performed according to a given taxonomy $\mathcal{T} = \langle \mathcal{C}, \leq \rangle$, where $C = \{c_k \mid k = 1, 2, \dots, N\}$ is a set of N predefined categories and “ \leq ” is a reflexive, anti-symmetric, and transitive binary relation.²

In the most general case, \mathcal{T} can be thought of as a strict partially ordered set (strict poset), which can be graphically represented by a DAG. We assume known all ordinary definitions concerning posets. However, for the sake of readability, let us recall some relevant definitions.

Covering Relation. Given a taxonomy $\mathcal{T} = \langle \mathcal{C}, \leq \rangle$, the covering relation “ \prec ” holds between comparable elements that are immediate neighbors in the taxonomy. In symbols: $b \prec a \Leftrightarrow b < a \wedge \neg \exists c \in \mathcal{C} \text{ s.t. } b < c < a$. The characteristic function $f : \mathcal{C} \times \mathcal{C} \rightarrow [0, 1]$ for the covering relation “ \prec ” is defined as:

$$f(b, a) = \begin{cases} 1 & \text{if } b \prec a \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

A “soft” version of the above definition would substitute “1” (used to denote full membership) with a number intended to measure to what extent the pair in question satisfies the covering relation. In a probabilistic setting, a natural choice for the characteristic function would be to let it coincide with the conditional probability $p(b|a)$. In symbols:

$$\forall a, b \in \mathcal{C} : b \prec a \iff f(b, a) \equiv p(b|a) > 0 \quad (2)$$

Ancestors, Offspring, and Children Sets. The notions of ancestors, offspring, and children sets, useful when dealing with taxonomies, can be easily defined for posets (hence, for DAGs and trees). Given a node $r \in \mathcal{C}$:

$$\begin{aligned} \mathcal{A}(r) &= \{a \in \mathcal{C} \mid r < a\} && \text{Ancestors set} \\ \mathcal{O}(r) &= \{o \in \mathcal{C} \mid o < r\} && \text{Offspring set} \\ \mathcal{H}(r) &= \{c \in \mathcal{C} \mid c \prec r\} && \text{Children set} \end{aligned} \quad (3)$$

Root, internal nodes, and leaves. A category without ancestors is called *root*; a category without children is called *leaf*, and a category with both ancestors and offspring is called *internal category*.

Two constraints must be effective for hierarchical text categorization:

² Some authors use “ $<$ ” instead of “ \leq ” as default binary relation. As the definition of “ $=$ ” and “ $<$ ” from “ \leq ” is trivial, in the following we will use “ $<$ ” when deemed useful for rendering definitions more intuitive.

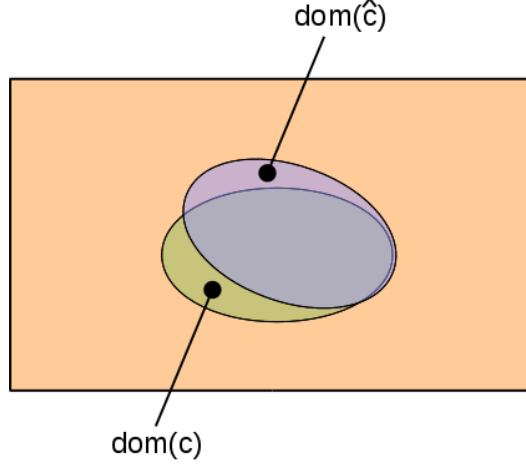


Figure 1: Overlapping between the domain of a category c and the domain of the corresponding classifier \hat{c} : the more overlapping, the better the behavior of the classifier is.

- *Hierarchical Consistency*. A label set $C_d \subseteq \mathcal{C}$ assigned to an instance $d \in D$ is said to be consistent with a given taxonomy $\mathcal{T} = \langle \mathcal{C}, \leq \rangle$ if it includes the complete ancestor sets for every label $c \in C_d$. In symbols: $c \in C_d \wedge b \in \mathcal{A}(c) \rightarrow b \in C_d$.
- *Hierarchical Consistency Requirement*. Any label assignments produced by a hierarchical classification system on a given categorization task has to be consistent with the underlying category taxonomy.

The notion of domain of a category c is also relevant, which denotes all documents that belong to c (i.e., the set of its positive instances).

Domain of a category. Given a taxonomy $\mathcal{T} = \langle \mathcal{C}, \leq \rangle$ the domain of a category $c \in \mathcal{C}$ is defined as: ³

$$\text{dom}(c) = \{i \mid i \triangleleft c \vee \exists a \in \mathcal{O}(c) \text{ s.t. } i \triangleleft a\} \quad (4)$$

We assume that each category $c \in \mathcal{C}$ embeds a corresponding binary classifier. Given an input, the classifier is entrusted with deciding whether or not it belongs to the corresponding category. To distinguish between a category and its embedded classifier, the latter will be denoted by a circumflex (i.e., \hat{c} denotes the classifier embedded by the category c).

The definition of domain can also be given for classifiers. In particular, $\text{dom}(\hat{c})$ denotes the set of inputs accepted by \hat{c} . In the ideal case in which $\text{dom}(\hat{c}) \equiv \text{dom}(c)$, we say that the classifier acts as an *oracle* for the given category. However, although a classifier is expected to approximate as much as possible the corresponding category, its domain typically does not coincide with that identified by the oracle (see Figure 1), i.e., $\text{dom}(\hat{c}) \neq \text{dom}(c)$.

³Where “ $i \triangleleft c$ ” denotes the *instance-of* relation that holds between the instance i and the category c .

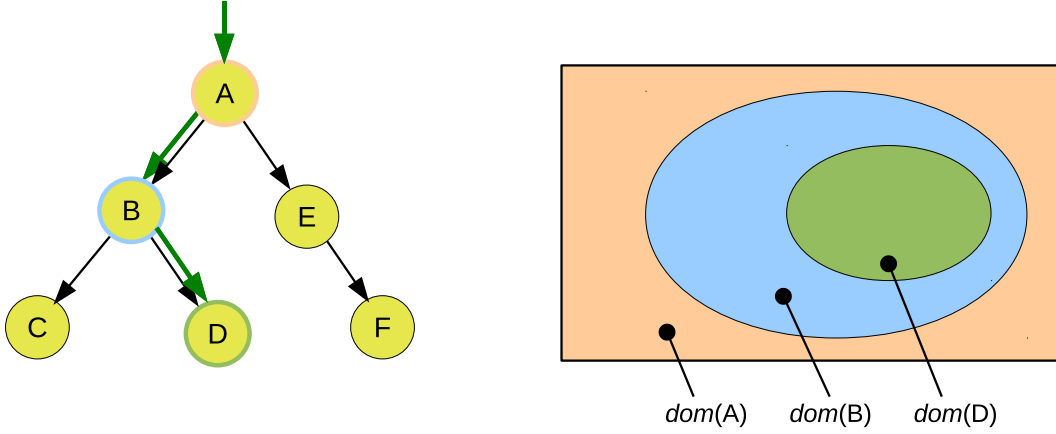


Figure 2: Graphical representation of the covering relation that holds among the domains of categories occurring along a path, e.g., from A to D through B.

Without loss of generality, we assume that the given taxonomy $\mathcal{T} = \langle \mathcal{C}, \leq \rangle$ has a unique root. In principle, the domains of categories that occur along a path originating from the root satisfy an inclusion relation (see Figure 2). The same kind of inclusion relation holds among the domains of the corresponding classifiers.

3.2 Non-Standard Definitions for HTC

We want \mathcal{T} to be represented by the set of all its most representative paths, i.e., those that originate from the root. Any one of these paths will be called *pipeline* hereinafter. Figure 3 depicts a simple source taxonomy, on the left part, and its “unfolding” in terms of pipelines, on the right part.

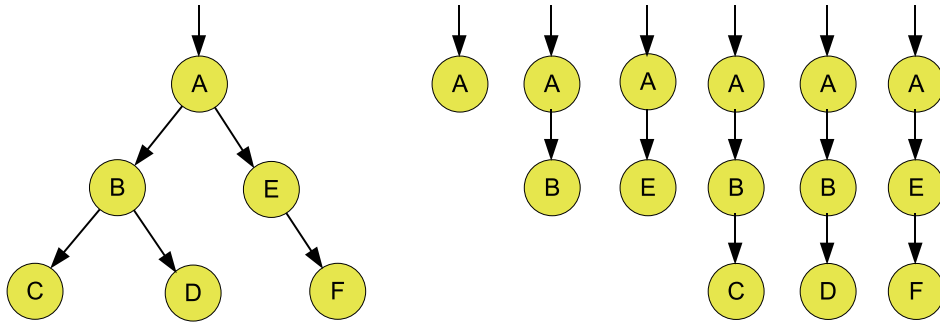


Figure 3: An example of taxonomy and its corresponding unfolding.

Well-Formed Strings and Pipelines. Given a taxonomy $\mathcal{T} = \langle \mathcal{C}, \leq \rangle$, a pipeline π is a well-formed string that originates from the root.

The definition of pipeline relies upon the concept of *well-formed string*, which in turn can be defined through the corresponding characteristic function $F : \mathcal{C}^* \rightarrow [0, 1]$:

$$F(w) = \begin{cases} 1 & w \equiv \lambda \vee w \in \mathcal{C} \\ F(\alpha) \cdot f(\beta, \alpha) \cdot F(\beta) & w = \alpha + \beta \equiv \alpha\beta, \alpha, \beta \in \mathcal{C}^+ \end{cases} \quad (5)$$

where:

- the operator “+” denotes concatenation between two strings (it can be omitted in absence of ambiguity);
- the constant λ denotes the empty string (with the property that $\forall \alpha \in \mathcal{C}^* : \alpha + \lambda \equiv \lambda + \alpha \equiv \alpha$);
- $f(\beta, \alpha)$ extends the characteristic function of the covering relation to pairs of strings in \mathcal{C}^+ , as follows: $f(\beta, \alpha) \equiv f(head(\beta), tail(\alpha))$, with *head* and *tail* having the usual semantic of extracting the first and the last element of their argument, respectively.

Note that, in a probabilistic setting, the characteristic function F represents the probability that a document will go through the corresponding pipeline under the assumption that the embedded classifiers act as oracles.

The set of well-formed strings \mathcal{W}_T and the set of pipelines \mathcal{P}_T in $\mathcal{T} = \langle \mathcal{C}, \leq \rangle$ can now be defined as follows:

$$\mathcal{W}_T = \{w \in \mathcal{C}^* \mid F(w) > 0\} \quad (6)$$

$$\mathcal{P}_T = \{\pi \in \mathcal{W}_T - \{\lambda\} \mid head(\pi) = root(\mathcal{T})\} \quad (7)$$

For instance, the path $A \rightarrow B \rightarrow D$ shown in Figure 2 gives rise to the string ABD , which is well-formed, as $D \prec B \prec A$, and rooted, as $head(ABD) = root(\mathcal{T})$; hence, it is a pipeline. A document of category D is expected to go through the pipeline ABD if correctly classified by the corresponding taxonomic system built upon \mathcal{T} .

A partial order also holds for pipelines. The concept we want to capture here is that an existing pipeline typically embeds other pipelines. In symbols:

$$\pi_1, \pi_2 \in \mathcal{P}_T : \pi_1 \leq \pi_2 \iff \exists w \in \mathcal{W}_T \text{ s.t. } \pi_2 = \pi_1 + w \quad (8)$$

Referring once again to Figure 3, other than the trivial assertion $ABC \leq ABC$, we can also state that $AB \leq ABC$ and that $A \leq ABC$ as $AB + C = A + BC = ABC$.

The reason why pipelines are considered so important lies in the fact that they facilitate the task of analyzing the corresponding taxonomy. In particular, pipelines can be extracted from both trees and DAGs, are immune from the problem of having to deal with multiple class labels, admit overlapping between class domains, and are naturally suited to deal with partial paths (a partial path originates from the root and terminates with an internal node of the taxonomy as opposed to a full path, which terminates with a leaf). Consequently, all issues that may arise depending on the characteristics of a hierarchical problem require the activation of suitable policies only

at the moment of moving from a pipeline-oriented to a taxonomy-oriented perspective, whereas the unfolding in terms of pipelines appears to be a common task for a variety of actual policies.

It is worth noting that the possibility of activating multiple paths within a taxonomy implies that, for at least one internal node $c \in \mathcal{C}$, an overlap occurs between (at least) two of its children. In symbols:

$$\exists a, b \in \mathcal{H}(c) : \text{dom}(a) \cap \text{dom}(b) \neq \emptyset \quad (9)$$

As for the assumption of having to deal with partial paths, this implies that at least one internal node $c \in \mathcal{C}$ has proper instances, not shared with any of its children. In symbols:

$$\text{dom}(c) \supset \bigcup_{a \in \mathcal{H}(c)} \text{dom}(a) \quad (10)$$

A further definition will be useful when discussing the main characteristics of PF. Assuming that $a, b \in \mathcal{C}$ and that $a \leq b$, the key concept we want to capture is that the relevant inputs for a with respect to b are all positive instances of b . To this end, let us define the notion of “relevance set” (*rset*) between two categories, as follows:

Relevance Set.

$$\forall a, b \in \mathcal{C} : \text{rset}(b, a) \triangleq \begin{cases} \text{dom}(a) & \text{if } b \leq a \\ \emptyset & \text{otherwise} \end{cases} \quad (11)$$

We can now check whether an input i is *relevant* (*rel*) for a category b , with respect to a category a , according to the definition below:

Relevance.

$$\text{rel}(i, b|a) \triangleq i \in \text{rset}(b, a) \quad (12)$$

4 Progressive Filtering

PF is a top-down strategy that requires the underlying taxonomy to be mirrored with local classifiers per node (hereinafter *LCN*), the underlying assumption being that the domain of a node/classifier encompasses the domains of its children. Consequently, classifiers are trained with hierarchical training sets and propagate an input only in the event that they accept it.⁴ This choice, together with the “pass-down” strategy, preserves hierarchical consistency, which imposes that all ancestors of a category that accepts an input must also accept it. As for the hierarchical consistency requirement, it may be satisfied or not, depending on the kind of structure in question (tree or

DAG) and on the policy adopted to deal with well-known issues, such as: (i) leaf node prediction, (ii) premature blocking, and (iii) high-level error recovering. We know that leaf-node prediction can be mandatory or not. In the latter case, for at least one input, its most specific class is not required to be a leaf node in the taxonomy, e.g., [29]. When the classification stops at an internal node while an oracle would keep propagating the current input downwards, then the blocking problem arises. Some strategies to avoid blocking are discussed, for instance, in [30]. The research community has also devoted efforts to cope with high-level error recovering. The interested reader will find several proposals aimed at tackling this issue in [8] and in [37].

4.1 Common solutions and known issues for PF

A common solution for implementing a binary classifier \hat{c} for a category $c \in \mathcal{C}$ consists of thresholding a real-valued classifier, entrusted with estimating the probability that an input belongs to c . In so doing, an optimization problem arises, which consists of identifying the threshold that maximizes/minimizes a utility/cost function, usually a well-known metric. The simplest solution to this problem, when classifiers are framed in a taxonomy, consists of independently optimizing pipelines of binary local classifiers, in which the *same* classifier is allowed to have different thresholds, depending on which pipeline it is embedded by, [1]. In so doing, a sort of “flattening” of the underlying taxonomy is performed, while pipelines still embed information about the underlying taxonomy.

PF can give rise to many other kinds of actual systems, depending on the given class of problems, on the choices made by the algorithm devised to solve them, and on the specific policies adopted to deal with the most well-known issues (see, for instance, [3]) encountered while trying to enforce the hierarchical consistency requirement.

It is worth noting that even simple scenarios may hide subtle issues. Just to give a taste of them, let us consider a case in which the given problem requires mandatory leaf-prediction. This implies, at least in principle, that any input accepted by an internal node c must be accepted by at least one of its children. As, by default, PF does not perform any direct action designed to enforce this property, the blocking problem may occur. Things deteriorate when one assumes that non-mandatory leaf-prediction is permitted, as nothing guarantees that stopping the acceptance of the current input at an internal node corresponds in fact to a correct categorization.

Summing up, a complete scenario of the probabilistic behavior of a taxonomy as a whole cannot be developed because of the large number of variations in terms of feasible policies and of the issues that may arise when trying to cope with the most well-known problems ensuing from a top-down strategy based on LCNs. Any specific solution (with its pros and cons) would generate a different statistical behavior, though still based on the pipelines extracted from the given taxonomy. This is the main reason why we concentrate on pipelines, which allow to perform a preliminary analysis regardless of the combination policy adopted. In particular, the focus will be first on classifiers in

⁴A dual strategy, not considered for PF, would assume that the domain of a node/classifier accounts only for its own inputs, disregarding the domains of its children. In this case, classifiers should be trained with proper training sets and propagate an input only in the event that they do not accept it.

isolation and then on pipelines of classifiers. In both cases, the concept of “normalized” confusion matrix is used to differentiate the probabilistic behavior of a classifier from the actual confusion matrices that summarize the results of specific experiments.

In the following, we also assume that the behavior of all classifiers is statistically significant. Under this assumption, we can model the outcome of a classifier embedded by a pipeline with two random variables, ranging over 0 (*false*) and 1 (*true*). In particular, following the choice made to distinguish oracles from actual classifiers, random variables related to oracles are denoted in plain format (e.g., X), whereas those related to actual classifiers have a circumflex (e.g., \hat{X}). Joint or conditional probabilities involved in the modeling activity, e.g., $p(X, \hat{X})$ and $p(\hat{X}|X)$, are represented with 2×2 matrices. Single random variables are also represented with 2×2 *diagonal* matrices, exploiting the fact that $p(X) \equiv p(X, X)$.

4.2 Analysis of a Single Classifier

Let us denote with $\Xi_c(p, n)$ the confusion matrix of a run in which a classifier \hat{c} embedded by a category $c \in \mathcal{C}$ is fed with m instances, of which p are positive and n negative. Paying attention to keeping the same values for p and n on different runs, the joint probability $p(X_c, \hat{X}_c)$ is proportional, through m , to the expected value of $\Xi_c(p, n)$. In symbols:

$$E[\Xi_c(p, n)] = m \cdot p(X_c, \hat{X}_c) \quad (13)$$

Assuming statistical significance, the confusion matrix obtained from a single test (or, better, averaged over multiple tests) gives us reliable information on the performance of a classifier. Hence, we can write:

$$\Xi_c(p, n) \approx m \cdot p(X_c, \hat{X}_c) = m \cdot p(X_c) \cdot p(\hat{X}_c|X_c) \quad (14)$$

We assume that the transformation performed by \hat{c} can be isolated from the inputs it processes, at least from a statistical perspective. In so doing, the confusion matrix for a given set of inputs can be written as the product between a term that accounts for the number of positive and negative instances, on the one hand, and a term that represents the expected recognition / error rate of \hat{c} . In symbols:

$$\Xi_c(p, n) = m \cdot \underbrace{\begin{bmatrix} \bar{f}_c & 0 \\ 0 & f_c \end{bmatrix}}_{\mathcal{O}(c) \approx p(X_c)} \cdot \underbrace{\begin{bmatrix} \gamma_{00} & \gamma_{01} \\ \gamma_{10} & \gamma_{11} \end{bmatrix}}_{\Gamma(c) \approx p(\hat{X}_c|X_c)} \quad (15)$$

where:

- $f_c = p/m$ and $\bar{f}_c = n/m$ denote the percent of positive and negative instances, respectively;

- $\gamma_{ij} \approx p(\hat{X}_c = j \mid X_c = i)$, $i, j = 0, 1$, denote the percent of inputs that have been correctly classified ($i = j$) or misclassified ($i \neq j$) by \hat{c} . In particular, γ_{00} , γ_{01} , γ_{10} , and γ_{11} denote the percent of true negatives (TN), false positives (FP), false negatives (FN), and true positives (TP), respectively. It can be easily verified that $\Gamma(c)$ is normalized row-by-row, i.e., that $\gamma_{00} + \gamma_{01} = \gamma_{10} + \gamma_{11} = 1$. For this reason, hereinafter an estimate of the conditional probability $p(\hat{X}_c \mid X_c)$ for a classifier \hat{c} embedded by a category c will be called *normalized confusion matrix*.

The separation between inputs and the intrinsic behavior of a classifier reported in Equation (15) suggests an interpretation that recalls the concept of transfer function, where a set of inputs is applied to \hat{c} . In fact, this could be interpreted alternatively as separating the optimal behavior of a classifier from the deterioration introduced by its actual filtering capabilities. In particular, $\mathcal{O}(c) \approx p(X_c)$ represents the *optimal behavior* obtainable when \hat{c} acts as an *oracle*, whereas $\Gamma(c) \approx p(\hat{X}_c \mid X_c)$ represents the *expected deterioration* caused by the actual characteristics of the classifier.

4.3 Analysis of a Pipeline of Classifiers

Pipelines are in fact the “building blocks” of the corresponding taxonomy. Without loss of generality, in the following we will adopt a naming scheme independent from the generic pipeline being investigated. In particular, the components of a pipeline π of length $L + 1$ are assumed to be the categories c_0, c_1, \dots, c_L (where c_0 represents the root), the underlying assumption being that $\forall k = 1, \dots, L : c_{k-1} \prec c_k$. An example of pipeline, extracted from a taxonomy and undergone to standard renaming, is shown in Figure 4.

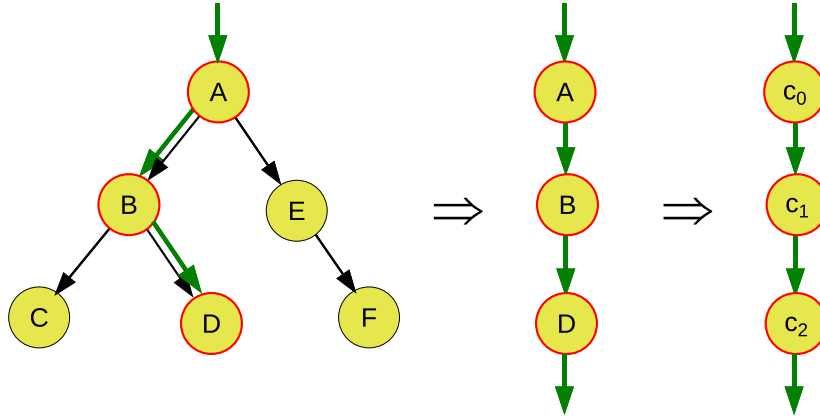


Figure 4: A pipeline of classifiers extracted from a taxonomy and undergone to standard renaming.

Let us also assume that $\pi_k \leq \pi$ denotes the “subpipeline” $c_0 c_1 \dots c_k$ and that $e(X, \hat{X})$ denotes co-occurring events involving an oracle and the corresponding classifier; in particular, e_{ij} will be used as a shorthand for $e(X = i, \hat{X} = j)$, $\forall i, j = 0, 1$. Still for the sake of readability, the domain of c_k will be denoted by A_k , whereas the domain of \hat{c}_k will be denoted by \hat{A}_k . The full list of shorthands defined with the goal of

simplifying the notation while deriving relevant formulas is reported in Table 1. Moreover, in absence of ambiguities, not-indexed quantities are meant to denote $k = L$, e.g., $\Omega_\pi(D) \equiv \Omega_{\pi_L}(D_L)$.

Studying classifiers embedded by a pipeline requires to model their interactions, which originate from the fact that the domain of a classifier \hat{c}_k is, by hypothesis, a proper subset of the domain of its ancestors. While the normalized confusion matrix of a classifier \hat{c} in isolation originates from $p(\hat{X}_c | X_c)$, additional conditions are required for a classifier embedded by a pipeline (except for the root), which accounts for the presence of its ancestors:

$$\Gamma^{(k)} \approx p(\hat{X}_k | X_k, \hat{X}_{k-1} = 1, \hat{X}_{k-2} = 1, \dots, \hat{X}_0 = 1) \quad (16)$$

However, due to the embedding of classifiers, some tautological implications imposed by the underlying taxonomy hold for $k > 0$ (see also the concept of “True Path Rule” in [32]):

$$X_{k-1} = 0 \models X_k = 0, \quad X_k = 1 \models X_{k-1} = 1 \quad (\text{from } A_k \subseteq A_{k-1}) \quad (17)$$

$$\hat{X}_{k-1} = 0 \models \hat{X}_k = 0, \quad \hat{X}_k = 1 \models \hat{X}_{k-1} = 1 \quad (\text{from } \hat{A}_k \subseteq \hat{A}_{k-1}) \quad (18)$$

Hence, considering that $\hat{X}_{k-1} = 1 \models \hat{X}_{k-2} = 1 \models \dots \models \hat{X}_0 = 1$, Equation (16) can be simplified as follows:

$$\Gamma^{(k)} \approx p(\hat{X}_k | X_k, \hat{X}_{k-1} = 1) \quad (19)$$

4.3.1 Finding an approximation for $p(X_k, \hat{X}_k)$

According to a probabilistic perspective, the starting point of our analysis is:

$$E[\Xi^{(k)}] = m \cdot p(X_k, \hat{X}_k) = m \cdot p(e^{(k)}) \quad (20)$$

As the process of estimating $p(X_k, \hat{X}_k)$ requires approximations, let us use a specific notation for the (estimation of) the joint probability $p(X_k, \hat{X}_k)$:

$$\Omega^{(k)} \approx p(e^{(k)}) \quad (21)$$

From the law of total probability, represented with the Bayes decomposition, each component of $\Omega^{(k)}$ can be represented as:

$$\omega_{ij}^{(k)} \approx p(e_{ij}^{(k)}) = \sum_{r,s} p(e_{rs}^{(k-1)}) \cdot p(e_{ij}^{(k)} | e_{rs}^{(k-1)}), \quad \forall i, j = 0, 1 \quad (22)$$

For the sake of brevity, we only derive $\omega_{00}^{(k)}$. The reader can consult APPENDIX A for further details on the derivation of $\omega_{ij}^{(k)}$, $\forall i, j = 0, 1$. To keep the notation simpler,

Table 1: Shorthands adopted while deriving relevant formulas.

Shorthand	Explanation
$\pi_k = \pi_{k-1} + c_k = c_0 c_1 \dots c_k$	Generic subpipeline ($k = 0, 1, \dots, L$), $\pi_k \leq \pi$
$\widehat{A}_k \triangleq \text{dom}(\widehat{c}_k), A_k \triangleq \text{dom}(c_k)$	Domains for \widehat{c}_k and the corresponding oracle c_k
\widehat{X}_k, X_k	Random variable for \widehat{c}_k and the corresponding oracle c_k
$e_{ij}^{(k)} \triangleq e(X_k = i, \widehat{X}_k = j)$	Co-occurring events, with $X_k = i$ and $\widehat{X}_k = j$
$f_k \triangleq p(X_k = 1 X_{k-1} = 1), \quad f_0 \triangleq 1$	Probability that an input in A_{k-1} also belongs to A_k
$\bar{f}_k \triangleq p(X_k = 0 X_{k-1} = 1) = 1 - f_k$	Complement of f_k
$F_k \triangleq p(X_k = 1) = \prod_{j=0}^k f_j$	Probability of traversing π_k , as classifiers were oracles
$\bar{F}_k \triangleq p(X_k = 0) = 1 - F_k$	Complement of F_k
$D_k = \{f_k j = 0, 1, \dots, k\}$	Set of conditional probabilities along π_k
$\Gamma^{(k)} \triangleq \Gamma(c_k)$	Normalized confusion matrix of the classifier \widehat{c}_k
$\Xi^{(k)} \triangleq \Xi_{\pi_k}(D_k; m)$	Confusion matrix for π_k , fed with m inputs
$\Omega^{(k)} \triangleq \Omega_{\pi_k}(D_k)$	Estimate of the joint probability $p(X_k, \widehat{X}_k)$
$\mathcal{O}^{(k)} \triangleq \mathcal{O}_{\pi_k}(D_k)$	Estimate of the prior probability $p(X_k)$
$\Phi^{(k)} \triangleq \Phi_{\pi_k}(D_k)$	Estimate of the conditional probability $p(\widehat{X}_k X_k)$

let us use “prime” to denote events or random variables that refer to the pipeline π_k , whereas plain text refers to π_{k-1} :

$$p(e'_{00}) = p(e_{00}) \cdot p(e'_{00}|e_{00}) + p(e_{01}) \cdot p(e'_{00}|e_{01}) + p(e_{10}) \cdot p(e'_{00}|e_{10}) + p(e_{11}) \cdot p(e'_{00}|e_{11})$$

where:

$$\begin{aligned} p(e'_{00}|e_{00}) &= p(X' = 0, \hat{X}' = 0 \mid X = 0, \hat{X} = 0) \\ &= \underbrace{p(\hat{X}' = 0 \mid X' = 0, X = 0, \hat{X} = 0)}_{=1} \cdot \underbrace{p(X' = 0 \mid X = 0, \hat{X} = 0)}_{=1} = 1 \\ p(e'_{00}|e_{01}) &= p(X' = 0, \hat{X}' = 0 \mid X = 0, \hat{X} = 1) \\ &= \underbrace{p(\hat{X}' = 0 \mid X' = 0, X = 0, \hat{X} = 1)}_{\approx \gamma'_{00}} \cdot \underbrace{p(X' = 0 \mid X = 0, \hat{X} = 1)}_{=1} \approx \gamma'_{00} \\ p(e'_{00}|e_{10}) &= p(X' = 0, \hat{X}' = 0 \mid X = 1, \hat{X} = 0) \\ &= \underbrace{p(\hat{X}' = 0 \mid X' = 0, X = 1, \hat{X} = 0)}_{=1} \cdot \underbrace{p(X' = 0 \mid X = 1, \hat{X} = 0)}_{\approx \bar{f}'} \approx \bar{f}' \\ p(e'_{00}|e_{11}) &= p(X' = 0, \hat{X}' = 0 \mid X = 1, \hat{X} = 1) \\ &= \underbrace{p(\hat{X}' = 0 \mid X' = 0, X = 1, \hat{X} = 1)}_{\approx \gamma'_{00}} \cdot \underbrace{p(X' = 0 \mid X = 1, \hat{X} = 1)}_{\approx \bar{f}'} \approx \gamma'_{00} \cdot \bar{f}' \end{aligned}$$

Hence:

$$p(e'_{00}) \approx \omega'_{00} = \omega_{00} + \omega_{01} \cdot \gamma'_{00} + \bar{f}' \cdot \omega_{10} + \bar{f}' \cdot \omega_{11} \cdot \gamma'_{00}$$

By making the derivation explicit for all $\omega_{ij}^{(k)}$, $i, j = 0, 1$, we can approximate $p(X_k, \hat{X}_k)$ as follows ($k > 0$):

$$\Omega^{(k)} = \begin{cases} \omega_{00}^{(k)} &= \omega_{00}^{(k-1)} + \omega_{01}^{(k-1)} \cdot \gamma_{00}^{(k)} + \bar{f}_k \cdot \omega_{10}^{(k-1)} + \bar{f}_k \cdot \omega_{11}^{(k-1)} \cdot \gamma_{00}^{(k)} \\ \omega_{01}^{(k)} &= 0 + \omega_{01}^{(k-1)} \cdot \gamma_{01}^{(k)} + 0 + \bar{f}_k \cdot \omega_{11}^{(k-1)} \cdot \gamma_{01}^{(k)} \\ \omega_{10}^{(k)} &= 0 + 0 + f_k \cdot \omega_{10}^{(k-1)} + f_k \cdot \omega_{11}^{(k-1)} \cdot \gamma_{10}^{(k)} \\ \omega_{11}^{(k)} &= 0 + 0 + 0 + f_k \cdot \omega_{11}^{(k-1)} \cdot \gamma_{11}^{(k)} \end{cases} \quad (23)$$

To help the reader better understand the underlying process, a graphical representation of the transformation that occurs along a pipeline from step $k-1$ to step k is given in Figure 5, which highlights how the elements of $\Omega^{(k-1)}$ concur to generate $\Omega^{(k)}$. Quantitative information, reported in Equation (23), is intentionally disregarded.

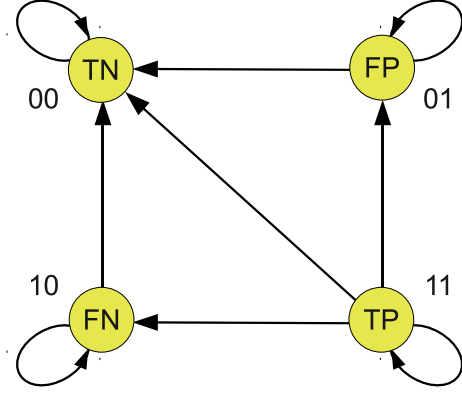


Figure 5: How the elements of $\Omega^{(k-1)}$ concur to generate $\Omega^{(k)}$. The presence of an arrow indicates that the source node (step $k - 1$) contributes to the destination node (step k). For instance, the arrow between TP and FP asserts that part of $\omega_{11}^{(k-1)}$ is responsible for $\omega_{01}^{(k)}$.

As for the base case (i.e., $k = 0$), it can be observed that the role of the root is to forward any incoming document down to its children. In other words, the (virtual) classifier embedded by the root accepts everything as a positive instance. For this reason, the base case for $\Omega^{(0)}$ is:

$$\Omega^{(0)} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad (24)$$

whereas the normalized confusion matrix of the root is:

$$\Gamma^{(0)} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \triangleq \mu \quad (25)$$

where μ is a constant that characterizes the *neutral classifier*, whose unique responsibility is to “pass everything down” to its children, no matter whether input documents are TP or FP.⁵

4.3.2 Revisiting One Step of Progressive Filtering

Looking at Equation (23), each processing step actually involves two separate actions. As sketched in Figure 6, everything goes as if the output of a classifier undergo *context switching* before *classification*.

Context switching. Concerns the fact that only part of TP output by \hat{c}_{k-1} are still TP for \hat{c}_k . Under the assumption of statistical significance (and recalling the definition of relevance set), the percent of relevant inputs for \hat{c}_k that move from TP to FP is

⁵Different choices could be made to represent the normalized confusion matrix of the root, without changing the result of the transformation that occurs there. However, the adoption of the neutral classifier appears the most intuitive. We will get back to this issue in the next subsection.

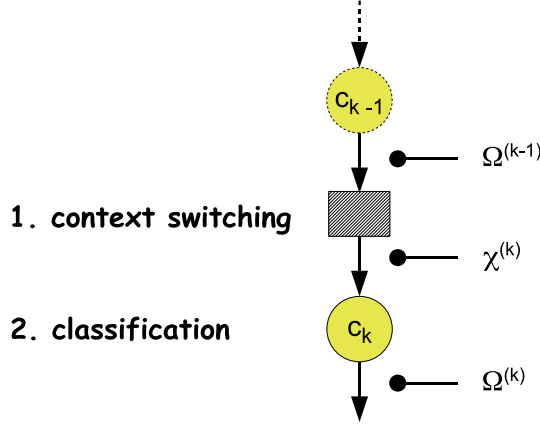


Figure 6: One step of progressive filtering.

approximately \bar{f}_k . Conversely, only part of FN output by \hat{c}_{k-1} are still FN for \hat{c}_k , so that the percent of inputs that move from false to TN is still \bar{f}_k . Hence, with χ and Ω representing the percent of inputs and the percent of outputs of a classifier in terms of true/false positives/negatives, we can write:

$$\chi^{(k)} = \begin{bmatrix} \omega_{00}^{(k-1)} + \bar{f}_k \cdot \omega_{10}^{(k-1)} & \omega_{01}^{(k-1)} + \bar{f}_k \cdot \omega_{11}^{(k-1)} \\ f_k \cdot \omega_{10}^{(k-1)} & f_k \cdot \omega_{11}^{(k-1)} \end{bmatrix} = \begin{bmatrix} 1 & \bar{f}_k \\ 0 & f_k \end{bmatrix} \cdot \Omega^{(k-1)} \quad (26)$$

Classification. The transformation performed by \hat{c}_k can be better understood highlighting that two paths can be followed by a document while going through the pipeline in hand: *inner* and *outer* path. Figure 7 illustrates the different paths followed by input documents while traversing a pipeline.

The *inner* path operates on true positives (χ_{11}) and false positives (χ_{01}). The corresponding transformation can be represented as follows:

$$\Omega^{(k)} \Big|_{inner} = \begin{bmatrix} 0 & \chi_{01}^{(k)} \cdot \gamma_{01}^{(k)} \\ 0 & \chi_{11}^{(k)} \cdot \gamma_{11}^{(k)} \end{bmatrix} = \begin{bmatrix} \chi_{01}^{(k)} & 0 \\ 0 & \chi_{11}^{(k)} \end{bmatrix} \cdot \begin{bmatrix} 0 & \gamma_{01}^{(k)} \\ 0 & \gamma_{11}^{(k)} \end{bmatrix} \quad (27)$$

The *outer* path operates on true negatives (χ_{00}) and false negatives (χ_{10}). The whole process is cumulative, and can be represented as follows (still for the classifier \hat{c}_k):

$$\Omega^{(k)} \Big|_{outer} = \begin{bmatrix} \chi_{00}^{(k)} + \chi_{01}^{(k)} \cdot \gamma_{00}^{(k)} & 0 \\ \chi_{10}^{(k)} + \chi_{11}^{(k)} \cdot \gamma_{10}^{(k)} & 0 \end{bmatrix} = \begin{bmatrix} \chi_{00}^{(k)} & 0 \\ \chi_{10}^{(k)} & 0 \end{bmatrix} + \begin{bmatrix} \chi_{01}^{(k)} & 0 \\ 0 & \chi_{11}^{(k)} \end{bmatrix} \cdot \begin{bmatrix} \gamma_{00}^{(k)} & 0 \\ \gamma_{10}^{(k)} & 0 \end{bmatrix} \quad (28)$$

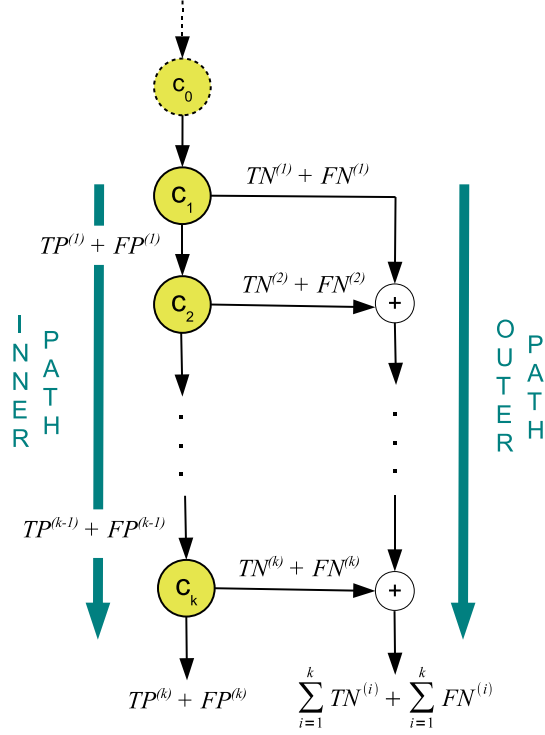


Figure 7: Inner and outer paths along a pipeline.

Putting together Equation (27) and (28), we obtain:

$$\Omega^{(k)} = \begin{bmatrix} \chi_{00}^{(k)} & 0 \\ \chi_{10}^{(k)} & 0 \end{bmatrix} + \begin{bmatrix} \chi_{01}^{(k)} & 0 \\ 0 & \chi_{11}^{(k)} \end{bmatrix} \cdot \Gamma^{(k)} \quad (29)$$

For its importance within the model, the transformation represented by Equation (29) deserves a specific definition.

Operator \oplus .

$$A \oplus B = \begin{bmatrix} \alpha_{00} & \alpha_{01} \\ \alpha_{10} & \alpha_{11} \end{bmatrix} \oplus \begin{bmatrix} \beta_{00} & \beta_{01} \\ \beta_{10} & \beta_{11} \end{bmatrix} \triangleq \begin{bmatrix} \alpha_{00} & 0 \\ \alpha_{10} & 0 \end{bmatrix} + \begin{bmatrix} \alpha_{01} & 0 \\ 0 & \alpha_{11} \end{bmatrix} \cdot \begin{bmatrix} \beta_{00} & \beta_{01} \\ \beta_{10} & \beta_{11} \end{bmatrix} \quad (30)$$

It is now easy to obtain a compact form for the transformation that occurs along the inner and the outer path of a pipeline. In symbols:

$$\Omega^{(k)} = \chi^{(k)} \oplus \Gamma^{(k)} = \underbrace{\left(\begin{bmatrix} 1 & \bar{f}_k \\ 0 & f_k \end{bmatrix} \cdot \Omega^{(k-1)} \right)}_{\text{context switching}} \oplus \underbrace{\Gamma^{(k)}}_{\text{classification}} \quad (31)$$

Note that Equation (31) can be applied also to the base case ($k = 0$), yielding:

$$\Omega^{(0)} = \chi^{(0)} \oplus \Gamma^{(0)} = \left(\begin{bmatrix} 1 & \bar{f}_0 \\ 0 & f_0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \right) \oplus \Gamma^{(0)} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \oplus \mu = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad (32)$$

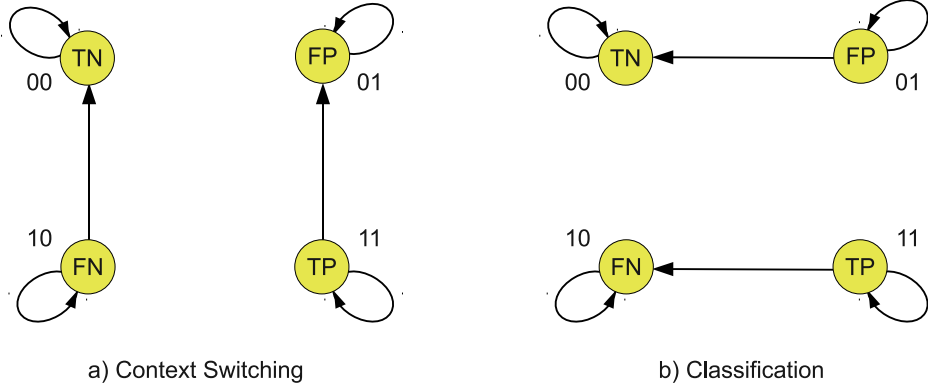


Figure 8: How the elements of $\Omega^{(k-1)}$ concur to generate $\Omega^{(k)}$, with separate focus for (a) context switching and (b) classification.

Equation (32) points out that neither the (virtual) context switching performed before submitting the input to the root nor the (virtual) processing of the root alter the given input –upon the assumption that $f_0 = 1$ (hence, $\bar{f}_0 = 0$) and that $\Gamma^{(0)} = \mu$.

Summarizing, the overall transformation can be represented as follows:

- Base case ($k = 0$), i.e., output of the root:

$$\Omega^{(0)} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad (33)$$

- Recursive step ($k > 0$), which coincides with Equation (23):

$$\Omega^{(k)} = \left(\begin{bmatrix} 1 & \bar{f}_k \\ 0 & f_k \end{bmatrix} \cdot \Omega^{(k-1)} \right) \oplus \Gamma^{(k)} \quad (34)$$

Figure 8 can help the reader better understand context switching and classification. As previously done, also in this case quantitative information is intentionally disregarded.

Unfolding the recurrence relation that defines Ω allows to obtain a closed formula, which accounts for the behavior of a pipeline π_k ($k > 0$):

$$\Omega^{(k)} = \begin{cases} \omega_{00}^{(k)} &= \bar{F}_k - \sum_{j=1}^k \bar{f}_j \cdot F_{j-1} \cdot \left(\prod_{r=0}^{j-1} \gamma_{11}^{(r)} \right) \cdot \left(\prod_{s=j}^k \gamma_{01}^{(s)} \right) \\ \omega_{01}^{(k)} &= \sum_{j=1}^k \bar{f}_j \cdot F_{j-1} \cdot \left(\prod_{r=0}^{j-1} \gamma_{11}^{(r)} \right) \cdot \left(\prod_{s=j}^k \gamma_{01}^{(s)} \right) \\ \omega_{10}^{(k)} &= F_k - F_k \cdot \prod_{j=0}^k \gamma_{11}^{(j)} \\ \omega_{11}^{(k)} &= F_k \cdot \prod_{j=0}^k \gamma_{11}^{(j)} \end{cases} \quad (35)$$

It is easy to verify from Equation (35) that $\omega_{00}^{(k)} = \bar{F}_k - \omega_{01}^{(k)}$ and that $\omega_{10}^{(k)} = F_k - \omega_{11}^{(k)}$; hence let us spend a few words to clarify the underlying semantics only for $\omega_{01}^{(k)}$ and $\omega_{11}^{(k)}$.

As for $\omega_{11}^{(k)}$, it represents the core behavior of PF. In particular, given an input, each classifier along the pipeline π_k accepts and forwards it with probability $f_j \cdot \gamma_{11}^{(j)}$, $j = 0, 1, \dots, k$. The resulting product can be split in two terms, one that accounts for the distribution of inputs and the other that accounts for the intrinsic properties of classifiers, as follows:

$$\prod_{j=0}^k (f_j \cdot \gamma_{11}^{(j)}) = \left(\prod_{j=0}^k f_j \right) \cdot \left(\prod_{j=0}^k \gamma_{11}^{(j)} \right) = F_k \cdot \left(\prod_{j=0}^k \gamma_{11}^{(j)} \right) \quad (36)$$

As for $\omega_{01}^{(k)}$, each component of the sum denotes a different subset of inputs, recognized as positive by \hat{c}_k but in fact negative. As these subsets are independent from each other, let us concentrate on a generic j -th element of the sum. In symbols:

$$\bar{f}_j \cdot F_{j-1} \cdot \left(\prod_{r=0}^{j-1} \gamma_{11}^{(r)} \right) \cdot \left(\prod_{s=j}^k \gamma_{01}^{(s)} \right) \quad j = 1, 2, \dots, k \quad (37)$$

Two processing modes hold along the pipeline π_k , and the switching occurs between \hat{c}_{j-1} and \hat{c}_j . Let us analyze these modes, together with the corresponding context switching:

- (a) *Processing mode before \hat{c}_j .* This behavior reproduces the one already analyzed for $\omega_{11}^{(k)}$, with the obvious difference that it is observed along the pipeline π_{j-1} ;
- (b) *Context switching between \hat{c}_{j-1} and \hat{c}_j .* The effect of context switching is to turn TP into FP, with probability \bar{f}_j ;

- (c) *Processing mode after \widehat{c}_{j-1} .* To keep “surviving” as FP, an input must be (incorrectly) recognized as positive by all the remaining classifiers that occur along the pipeline, including \widehat{c}_j , each with probability $\gamma_{01}^{(s)}$, $s = j, j+1, \dots, k$.

According to the ordering followed by the enumeration above, Equation (37) can be rewritten as:

$$\underbrace{F_{j-1} \cdot \left(\prod_{r=0}^{j-1} \gamma_{11}^{(r)} \right)}_{(a)} \cdot \underbrace{\bar{f}_j}_{(b)} \cdot \underbrace{\left(\prod_{s=j}^k \gamma_{01}^{(s)} \right)}_{(c)} \quad j = 1, 2, \dots, k \quad (38)$$

Now that the semantics of all elements reported in $\Omega^{(k)}$ has been clarified, let us try to give $\Omega^{(k)}$ a more concise form through the following definitions:

$$\prod_{i=0}^k \gamma_{01}^{(i)} \triangleq \psi_{01}^{(k)}, \quad \prod_{i=0}^k \gamma_{11}^{(i)} \triangleq \psi_{11}^{(k)}, \quad \text{and} \quad \frac{1}{F_k} \cdot \sum_{i=1}^k \bar{f}_i \cdot F_{i-1} \cdot \frac{\psi_{11}^{(i-1)}}{\psi_{01}^{(i-1)}} \triangleq \eta^{(k)} \quad (39)$$

According to these definitions, $\Omega^{(k)}$ can be rewritten as ($k > 0$):

$$\Omega^{(k)} = \underbrace{\begin{bmatrix} \bar{F}_k & 0 \\ 0 & F_k \end{bmatrix}}_{\mathcal{O}^{(k)}} \cdot \underbrace{\begin{bmatrix} 1 - \eta^{(k)} \cdot \psi_{01}^{(k)} & \eta^{(k)} \cdot \psi_{01}^{(k)} \\ 1 - \psi_{11}^{(k)} & \psi_{11}^{(k)} \end{bmatrix}}_{\Phi^{(k)}} \quad (40)$$

where $\mathcal{O}^{(k)}$ accounts for the optimal behavior of the pipeline π_k (as all its classifiers were oracles), whereas $\Phi^{(k)}$ represents the expected deterioration, due to the actual behavior of π_k .

It is easy to verify that Φ , which plays for pipelines the role that Γ plays for single classifiers, is also normalized row-by-row for each $k = 1, 2, \dots, L$. As for $\Phi^{(0)}$, we know that the following equivalence must hold: $\Phi^{(0)} = \Gamma^{(0)} = \mu$. Hence, as expected, also $\Phi^{(0)}$ is normalized row-by-row.

Moreover, as $\Omega^{(k)}$ spans over the whole space of events, the sum over its components must be 1. While trivially true for $k = 0$, it is easy to show it for any $k > 0$. Starting from Equation (40), we can write:

$$\sum_{ij} \omega_{ij} = \bar{F}_k \cdot \phi_{00}^{(k)} + \bar{F}_k \cdot \phi_{01}^{(k)} + F_k \cdot \phi_{11}^{(k)} + F_k \cdot \phi_{10}^{(k)} = \bar{F}_k + F_k = 1 \quad (41)$$

Let us also note that $\eta^{(k)}$ is only apparently not defined when $\psi_{01}^{(j-1)} \equiv 0$, for some $j > 1$. For instance, assuming that an index i exists such that $\gamma_{01}^{(i)} = 0$, we have

$\forall j > i : \psi_{01}^{(j-1)} = 0$, which in turn implies that $\forall j > i : \eta^{(j)} = \infty$. However, $\omega_{01}^{(k)}$ (and thus $\omega_{00}^{(k)}$) is *still* defined for any $k \geq 0$, as:

$$\omega_{01}^{(k)} = \lim_{\gamma_{01}^{(i)} \rightarrow 0} \bar{F}_k \cdot \eta^{(k)} \cdot \psi_{01}^{(k)} \equiv \sum_{j=1}^k \bar{f}_j \cdot F_{j-1} \cdot \psi_{11}^{(k)} \cdot \prod_{s=j}^k \gamma_{01}^{(s)} = \sum_{j=i+1}^k \bar{f}_j \cdot F_{j-1} \cdot \psi_{11}^{(k)} \cdot \prod_{s=j}^k \gamma_{01}^{(s)}$$

Hence, the confusion matrix Ξ for a pipeline of classifiers π , to which m inputs with known conditional probabilities D (with reference to the categories involved in the pipeline) are applied, can always be represented as:

$$\Xi_{\pi}(D; m) = m \cdot \Omega_{\pi}(D) = m \cdot \underbrace{\begin{bmatrix} \bar{F} & 0 \\ 0 & F \end{bmatrix}}_{\mathcal{O}_{\pi}(D)} \cdot \underbrace{\begin{bmatrix} 1 - \eta \cdot \psi_{01} & \eta \cdot \psi_{01} \\ 1 - \psi_{11} & \psi_{11} \end{bmatrix}}_{\Phi_{\pi}(D)} \quad (42)$$

It is now clear that Ω and Φ depend both on the conditional probabilities that characterize the flow of inputs along the pipeline (through η) and on the characteristics of the involved classifiers (through ψ_{01} and ψ_{11}). However, ψ_{01} and ψ_{11} depend only on the *intrinsic properties* of the classifiers involved in a pipeline, and are in fact building blocks for defining Φ and Ω . In the following subsection, we better analyze this issue.

4.4 Intrinsic Properties of a Pipeline

A recursive definition for ψ_{01} and ψ_{11} (actually, for the matrix Ψ) can be easily given in terms of the “ \oplus ” operator, as follows:

Definition of Ψ .

$$\Psi^{(k)} = \begin{cases} \mu & k = 0 \\ \Gamma^{(1)} & k = 1 \\ \Psi^{(k-1)} \oplus \Gamma^{(k)} & k > 1 \end{cases} \quad (43)$$

Where the choice of reporting the base case with $k = 1$ has been introduced only for the sake of readability, as it is consistent with the base case with $k = 0$. In symbols:

$$\Psi^{(1)} = \Psi^{(0)} \oplus \Gamma^{(1)} = \mu \oplus \Gamma^{(1)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \Gamma^{(1)} \equiv \Gamma^{(1)} \quad (44)$$

Note that the row-by-row normalization property is preserved also for Ψ , no matter how many classifiers occur in the pipeline. We can verify it by induction from Equation (43): with $\Gamma^{(k)}$ normalized by definition and assuming that $\Psi^{(k-1)}$ is normalized, we only need to verify that $\Psi^{(k)}$ preserves this property. To show it, let us rewrite Equation (43) as follows ($k > 0$):

$$\begin{aligned} \psi_{00}^{(k)} &= \psi_{00}^{(k-1)} + \psi_{01}^{(k-1)} \cdot \gamma_{00}^{(k)} & \psi_{01}^{(k)} &= \psi_{01}^{(k-1)} \cdot \gamma_{01}^{(k)} \\ \psi_{10}^{(k)} &= \psi_{10}^{(k-1)} + \psi_{11}^{(k-1)} \cdot \gamma_{10}^{(k)} & \psi_{11}^{(k)} &= \psi_{11}^{(k-1)} \cdot \gamma_{11}^{(k)} \end{aligned}$$

Summing up row-by-row:

$$\begin{aligned}\psi_{00}^{(k)} + \psi_{01}^{(k)} &= \psi_{00}^{(k-1)} + \psi_{01}^{(k-1)} \cdot \left(\gamma_{00}^{(k)} + \gamma_{01}^{(k)} \right) = \psi_{00}^{(k-1)} + \psi_{01}^{(k-1)} = 1 \\ \psi_{10}^{(k)} + \psi_{11}^{(k)} &= \psi_{10}^{(k-1)} + \psi_{11}^{(k-1)} \cdot \left(\gamma_{10}^{(k)} + \gamma_{11}^{(k)} \right) = \psi_{10}^{(k-1)} + \psi_{11}^{(k-1)} = 1\end{aligned}$$

Unfolding the definition of $\Psi^{(k)}$ and taking into account the normalization property we can write ($k \geq 0$):

$$\Psi^{(k)} = \begin{bmatrix} \sum_{j=1}^k \gamma_{00}^{(j)} \cdot \prod_{i=0}^{j-1} \gamma_{01}^{(i)} & \prod_{i=0}^k \gamma_{01}^{(i)} \\ \sum_{j=1}^k \gamma_{10}^{(j)} \cdot \prod_{i=0}^{j-1} \gamma_{11}^{(i)} & \prod_{i=0}^k \gamma_{11}^{(i)} \end{bmatrix} = \begin{bmatrix} 1 - \prod_{j=0}^k \gamma_{01}^{(k)} & \prod_{j=0}^k \gamma_{01}^{(k)} \\ 1 - \prod_{j=0}^k \gamma_{11}^{(k)} & \prod_{j=0}^k \gamma_{11}^{(k)} \end{bmatrix} \quad (45)$$

It is worth pointing out that Ψ , which gives the expected result for ψ_{01} and ψ_{11} , can be seen as a relaxed form of Φ , in which –for a pipeline π_k – all negative inputs are taken *outside* the domain of c_1 whereas all positive inputs are taken *inside* the domain of c_k . This choice can be imposed in the model of Ω by setting $0 < f_1 < 1$ and $f_j = 1$, $j = 2, \dots, k$ (f_0 is always equal to 1, by definition), so that $\eta^{(k)}$, F_k and \bar{F}_k reduce to 1, f_1 and \bar{f}_1 , respectively. This implies that no adaption is required for classifiers in the pipeline, except for \hat{c}_1 . Under this restrictive hypothesis, $\Omega^{(k)}$ reduces to ($k > 0$):

$$\Omega^{(k)} = \underbrace{\begin{bmatrix} \bar{f}_1 & 0 \\ 0 & f_1 \end{bmatrix}}_{\mathcal{O}^{(k)}} \cdot \underbrace{\begin{bmatrix} 1 - \psi_{01}^{(k)} & \psi_{01}^{(k)} \\ 1 - \psi_{11}^{(k)} & \psi_{11}^{(k)} \end{bmatrix}}_{\Phi^{(k)} \equiv \Psi^{(k)}} \quad (46)$$

As Equation (46) accounts *only* for the internal structure of the corresponding pipeline, one can hypothesize that Ψ is in fact a homomorphism which maps elements from \mathcal{C}^* to the space of normalized confusion matrices, say $\mathcal{M} \equiv [0, 1]^4$. In symbols:

$$\Psi : \mathcal{C}^* \rightarrow \mathcal{M} \quad (47)$$

Indeed, given a taxonomy $\mathcal{T} = \langle \mathcal{C}, \leq \rangle$, it is easy to verify that Ψ is a homomorphism, as:

- the Kleene star of \mathcal{C} yields in fact a monoid, closed under the concatenation operation (denoted with “+”), associative, and whose neutral element is the empty string λ ;
- the space \mathcal{M} of normalized confusion matrices is also a monoid, closed under the “ \oplus ” operation, associative, and whose neutral element is the neutral classifier μ .

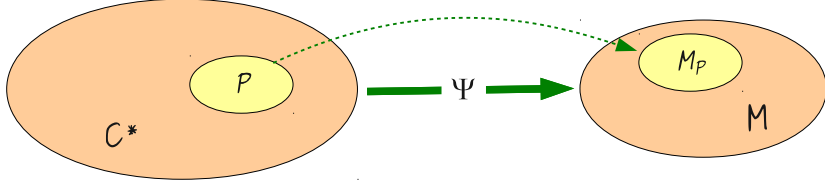


Figure 9: The homomorphism Ψ , which holds between \mathcal{C}^* and \mathcal{M}

This is due to the fact that Ψ preserves the structure, with “ \oplus ” and μ playing in \mathcal{M} the role that “ $+$ ” and λ play in \mathcal{C}^* . The interested reader can consult APPENDIX B for further details concerning this issue.

Note that, although Ψ is defined for any string in \mathcal{C}^* , we are in fact interested in pipelines (see Figure 9). However, as already pointed out, they can be easily identified throughout the characteristic function $F : \mathcal{C}^* \rightarrow [0, 1]$, which is strictly greater than zero only for well-formed strings, with the additional constraint that, to be pipelines, they must originate from the root.

Moreover, thanks to the associative property, it is also possible to give constructive definitions for \mathcal{M} through Ψ . In symbols (with $c \in \mathcal{C}$, $\pi \in \mathcal{C}^*$, and $\Psi(\lambda) \equiv \mu$):

- *Right recursion*: $\pi' = c + \pi \Rightarrow \Psi(\pi') = \Psi(c + \pi) = \Psi(c) \oplus \Psi(\pi) \equiv \Gamma(c) \oplus \Psi(\pi)$
- *Left recursion*: $\pi' = \pi + c \Rightarrow \Psi(\pi') = \Psi(\pi + c) = \Psi(\pi) \oplus \Psi(c) \equiv \Psi(\pi) \oplus \Gamma(c)$

5 Analysis on Relevant Metrics

5.1 Taxonomic Variations of Ordinary Metrics

According to the focus of the paper, we propose straightforward definitions for precision, recall, and $F1$. To differentiate them from other proposals (e.g., hP , hR , and $hF1$ defined in [20]), they will be denoted as tP , tR , and $tF1$ –standing for “taxonomic” P , R , and $F1$, respectively. Their definition apply to pipelines and strictly follow the probabilistic modeling represented by the Ω matrix. As the elements of $\Omega^{(k)}$ can be used to represent the overall transformation performed by π_k , calculating tP , tR , and $F1$ for a pipeline π_k is now straightforward:

$$tP(\pi_k) = \frac{\omega_{11}^{(k)}}{\omega_{11}^{(k)} + \omega_{01}^{(k)}} = \left(1 + \frac{\omega_{01}^{(k)}}{\omega_{11}^{(k)}}\right)^{-1} = \left(1 + \eta^{(k)} \cdot \frac{\bar{F}_k}{F_k} \cdot \frac{\psi_{01}^{(k)}}{\psi_{11}^{(k)}}\right)^{-1} \quad (48)$$

$$tR(\pi_k) = \frac{\omega_{11}^{(k)}}{\omega_{11}^{(k)} + \omega_{10}^{(k)}} = \frac{F_k \cdot \psi_{11}^{(k)}}{F_k \cdot \psi_{11}^{(k)} + F_k \cdot \psi_{10}^{(k)}} = \psi_{11}^{(k)} \quad (49)$$

$$tF1(\pi_k) = 2 \cdot \left(\frac{1}{P} + \frac{1}{R}\right)^{-1} = 2 \cdot \left[\left(1 + \eta^{(k)} \cdot \frac{\bar{F}_k}{F_k} \cdot \frac{\psi_{01}^{(k)}}{\psi_{11}^{(k)}}\right) + \left(\frac{1}{\psi_{11}^{(k)}}\right)\right]^{-1} \quad (50)$$

As for the taxonomic accuracy (tA), although less important for assessing the behavior of classifiers in pipeline (mainly due to the fact that usually the imbalance between

positive and negative inputs rapidly grows with the depth level of the classifier under analysis), it can be easily defined as well:

$$tA(\pi_k) = \frac{\omega_{00}^{(k)} + \omega_{11}^{(k)}}{\sum_{ij} \omega_{ij}^{(k)}} = tr(\Omega^{(k)}) = \bar{F}_k \cdot \left(1 - \eta^{(k)} \cdot \psi_{01}^{(k)}\right) + F_k \cdot \psi_{11}^{(k)} \quad (51)$$

where $tr(\cdot)$ denotes the *trace* of a matrix, obtained by summing up the elements of its main diagonal.

5.2 How Taxonomic Precision, Recall, and F_1 change along a pipeline

To better assess PF , let us analyze how the most relevant metrics change along a pipeline π . Our analysis proceeds by induction, assuming of having assessed the behavior of π_{k-1} , and then verifying what happens if one adds a further classifier.⁶ Again for the sake of readability, let us distinguish any relevant parameter concerning the pipelines π_k and π_{k-1} with a “prime” and a “plain” notation, respectively (for instance, F' denotes F_k , whereas F denotes F_{k-1}).

- *Precision* – Imposing that $tP(\pi') - tP(\pi) \geq 0$, the constraint on tP that involves the relevant parameters of \hat{c}_k is:

$$\gamma'_{01} \leq \frac{\bar{F}\eta}{\bar{F}'\eta'} \cdot f' \cdot \gamma'_{11} \quad (52)$$

where:

$$\bar{F}'\eta' = \sum_{j=1}^k \bar{f}_j \cdot F_{j-1} \cdot \frac{\psi_{11}^{(k)}}{\psi_{01}^{(j-1)}} = \bar{F}\eta + \bar{f}' \cdot F \cdot \frac{\psi_{11}}{\psi_{01}} > \bar{F}\eta$$

Summarizing, tP may increase or not along a pipeline depending on the constraint reported in Equation (52), which is strictly related with the behavior of the ratio $\bar{F}\eta/\bar{F}'\eta'$. Note that the behavior of tP depends on the distribution of data expected to flow along the pipeline.

- *Recall* – Imposing that $tR(\pi') - tR(\pi) \geq 0$, the constraint on tR that involves the relevant parameters of the classifier \hat{c}_k is:

$$\gamma'_{11} \geq 1 \quad (53)$$

It is clear that the constraint on tR is satisfied only when $\gamma'_{11} = 1$. Hence, tR is *monotonically decreasing* along a pipeline, the lower γ'_{11} (i.e., the percent of TP) the greater the decrement of R . Note that the behavior of tR does not depend on the distribution of data expected to flow along the pipeline.

⁶ As, by definition, $P(\pi_0) = R(\pi_0) = F1(\pi_0) \equiv 1$, the constraints for π_1 are in fact not relevant. For this reason, the analysis concerns only pipelines π_k with $k > 1$.

- F_1 – According to the given definition, tF_1 lies in between tP and tR . It is typically decreasing, unless the expected behavior of tR (monotonically decreasing) is more than counterbalanced by an increase of tP .

6 Discussion

Two main questions have been formulated at the beginning of the paper (Section 1), concerning (i) the possibility of predicting the expected behavior of a system implemented in accordance with PF when fed with a corpus of documents whose statistical properties are known and (ii) the possibility of separating the statistical information concerning inputs from the intrinsic properties of the classifiers embedded in the given taxonomy. After focusing on the above questions, the discussion will also summarize the analysis performed on relevant metrics.

6.1 Predicting the Expected behavior of a PF System

We have shown that it is very difficult to analyze a taxonomy as a whole, also due to the number of variants that can be put into practice while trying to enforce the hierarchical consistency requirement. Rather, it becomes surprisingly easy upon the extraction of the corresponding set of pipelines.

As the process of unfolding a taxonomy can be put into practice in many different scenarios, the analysis in terms of pipelines is apparently a common step for any LCN approach, including PF. Indeed, the unfolding process does not require specific constraints to be satisfied by the problem in hand. In particular, it can be performed in presence of (i) trees or DAGs, (ii) non-overlapping or overlapping among (the domains of) categories, and (iii) mandatory or non-mandatory leaf-node prediction.

Starting from the assumption that the confusion matrix measured after performing an experiment with a pipeline $\pi = c_0 c_1 \dots c_L$ is in fact a single realization of a probabilistic process, the following equation holds (see § 4.3):

$$\Xi_\pi(D; m) = m \cdot \Omega_\pi(D) \approx m \cdot p(X_L, \hat{X}_L) \quad (54)$$

where $\Omega_\pi(D)$ accounts for the behavior of π from a probabilistic perspective, as it is an estimation of the joint probability $p(X_L, \hat{X}_L)$. An *effective procedure* for evaluating Ω has been given, according to the knowledge about the behavior of the classifiers embedded by the pipeline, represented by their normalized confusion matrices $\Gamma(c_k)$, $k = 0, 1, \dots, L$, and about the expected distribution of inputs. Summarizing, the answer to the first question is positive, as one can easily use the analysis performed in terms of pipelines to predict the behavior of a hierarchical system compliant with PF. Note that the analysis can be performed only when the distribution of data is known, otherwise the model will not approximate well the real-world. However, for large scale data, e.g., web applications that process user queries, the hypothesis of knowing the distribution of data is not difficult to fulfill.

6.2 Separating the Statistical Information Concerning Inputs from the Intrinsic Properties of Classifiers

We have shown (§ 4.3) that $\Omega_\pi(D)$ can be represented as the product between $\mathcal{O}_\pi(D)$ and $\Phi_\pi(D)$. Considering that $\Omega_\pi(D)$ approximates the joint probability $p(X_L, \hat{X}_L)$, we can write:

$$p(X_L, \hat{X}_L) \approx \Omega_\pi(D) = \mathcal{O}_\pi(D) \cdot \Phi_\pi(D) \quad (55)$$

where $\mathcal{O}_\pi(D) \approx p(X_L)$ denotes the behavior of a pipeline under the hypothesis that all classifiers it embeds were acting as oracles, whereas $\Phi_\pi(D) \approx p(\hat{X}_L|X_L)$ represents the expected deterioration. We have pointed out that the Φ plays for pipelines the role that Γ plays for single classifiers. However, although the property of row-by-row normalization is satisfied for both Φ and Γ , Φ *still* depends on the distribution of input data while Γ does not. Fortunately, some building blocks have been identified, characterized by the Ψ matrix, whose elements depend *only* on the intrinsic properties of the pipeline. The dependence of Φ from Ψ is highlighted by the following formula, which is very important in the process of pipeline analysis:

$$\Omega_\pi(D) = \underbrace{\begin{bmatrix} \bar{F} & 0 \\ 0 & F \end{bmatrix}}_{\mathcal{O}_\pi(D)} \cdot \underbrace{\begin{bmatrix} 1 - \eta \cdot \psi_{01} & \eta \cdot \psi_{01} \\ 1 - \psi_{11} & \psi_{11} \end{bmatrix}}_{\Phi_\pi(D)} \quad (56)$$

Note that, due to its independence from the distribution of data, the task of calculating Ψ can be done once, and requires to be repeated only in the event that the properties of one or more classifiers in the pipeline change. Summarizing, the answer here is only partially positive, as the approximated model represented by Ω cannot be expressed in a way that clearly separates the distribution of input data (through \mathcal{O}) from the intrinsic behavior of the pipeline (through Φ). In fact, Φ still embeds the information about the distribution of input data. However, one can calculate $\Psi(\pi)$ for each pipeline $\pi \in \mathcal{P}_T$, starting from the normalized confusion matrices Γ of the classifiers embedded by π . The Ψ matrices are independent from the labeling of the input data, and can be used, together with the set of conditional probabilities that characterize the inclusion relations for the given pipeline, to calculate the normalized confusion matrix of the pipeline (i.e., Φ) and therefore the approximated model (i.e., Ω).

As a noticeable consequence of Equation (56), testing the behavior of a pipeline for a specific value of imbalance is not straightforward. The motivation lies in the fact that the same imbalance can be obtained with many different distributions of inputs. To better highlight this issue, let us assume that one wants to measure the behavior of a pipeline in presence of 10% of positive vs. 90% of negative inputs. Positive inputs refer to the last classifier in the pipeline, and their amount is fixed (in this case, 10%). On the other hand, negative inputs can be selected in a variety of ways along the pipeline. For instance, one may select only inputs that do not belong to any category but the root (which by hypothesis accepts all inputs).⁷ Another peculiar policy may consist of

selecting as negative inputs only those that belong to the last but one classifier in the pipeline. However, the above policies for negative input selection are not representative enough for identifying the behavior of a pipeline in presence of imbalance. In fact, many other selection policies are feasible, provided that the constraint on imbalance is satisfied. Summarizing, while the problem of setting up test beds with statistical significance remains (no matter whether the corresponding tests are performed with a single run, averaging over multiple runs, or resorting to k -fold cross validation), a further problem arises for pipeline testing, as its behavior depends on the distribution of inputs being processed. Hence, studying the imbalance requires at least an averaging over multiple test, each run with a different distribution of (negative) inputs.

6.3 Analysis Performed on Relevant Metrics

The analysis performed on relevant metrics (i.e., tP , tR , tF_1) highlights that tP depends on the distribution of data while tR does not. As for tR , we have shown that it is monotonically decreasing. This phenomenon is related with the problem of high-level error recovering, which originates from the fact that errors made at higher levels of a taxonomy have great impact on the overall performance of any actual system that implements top-down processing (including those based on PF). The impact of high-level errors on the overall performance of a system can be better understood recalling the concepts of inner and outer path: the former is entrusted with performing progressive filtering, whereas the latter accumulates inputs that have been rejected by any of the classifiers embedded by the pipeline. For this reason, there is no way to recover errors performed along the outer path (FN), while errors performed by a classifier along the inner path (FP) may be recovered by subsequent processing steps. This behavior is also highlighted by the study made on relevant metrics, where the recall (related to FN) is monotonically decreasing, whereas the precision (related to FP) may be increasing or not depending on the characteristics of the involved classifiers. A simple strategy headed to limit the impact of high-level errors can be put into practice by lowering the thresholds of the embedded classifiers, the closer the classifier to the root, the lower the threshold. In so doing, FN are expected to decrease while FP are expected to increase. However, FP can be further processed by the classifiers that occur after the current one in the given pipeline, thus literally realizing “progressive filtering”. This strategy affects also the training of classifiers, which are required to maintain the same discrimination capabilities on relevant and non relevant inputs that originate from their ancestors (see the definition of relevant input given in Section 3). The main consequence of relaxing the behavior of \hat{c}_{k-1} (more generally, of the pipeline π_{k-1}) is that the set of relevant inputs for \hat{c}_k is extended with FP that originate from \hat{c}_{k-1} and its ancestors. Hence, the training activity should be performed taking into account this issue, with the goal of improving the robustness of \hat{c}_k towards non-relevant FP.

⁷This issue has already been described in § 4.4, while defining the Ψ matrix.

7 Conclusions and Future Work

In this paper, a formal modeling of the progressive filtering technique has been performed, according to a probabilistic perspective and framed within the research field of hierarchical text categorization. In particular, the focus has been on how to assess pipelines extracted from a given taxonomy. Specific care has been taken in identifying whether some building blocks exist in the model, which are independent from the underlying distribution of input data. This part of the analysis has brought to the definition of the Ψ matrix, which accounts only for the structural aspects of a pipeline. How to separate the expected optimal behavior of a pipeline from the deterioration introduced by the actual classifiers it embeds is another important result. The way relevant metrics change along a pipeline has also been investigated. As expected, the precision may increase or decrease depending on the characteristics of the embedded classifiers, whereas the recall is monotonically decreasing along a pipeline. To limit the impact of this latter unwanted behavior, one may relax the behavior of classifiers at higher levels, thus reducing the overall percent of FN. The results of the analysis performed in this paper should facilitate the designer of a system based on progressive filtering in the task of devising, training and testing it. In particular, some relevant scenarios have been sketched in Section 1, in which the proposed probabilistic model can be useful.

As for future work, we are currently investigating the problem of which policy should be applied to train classifiers embedded in a taxonomy. Moreover, we are about to use the model in a problem of threshold optimization.

Acknowledgements. Many thanks to all people that gave me help in the task of conceptualizing and formalizing this work. Special thanks go to Eloisa Vargiu, who first conjectured the possibility that part of the FN disregarded by a classifier should be turned into TN, and to Mauro Parodi, for his wealth of ideas about linear transformations and their application to model text categorization tasks in a hierarchical setting. Many thanks also to Donato Malerba, Giorgio Valentini, and Fabio Roli, who made a preliminary review of the manuscript.

References

- [1] Addis, A., Armano, G., Vargiu, E.: Assessing Progressive Filtering to Perform Hierarchical Text Categorization in Presence of Input Imbalance, *Proceedings of International Conference on Knowledge Discovery and Information Retrieval (KDIR 2010)*, 2010.
- [2] Barutcuoglu, Z., Schapire, R. E., Troyanskaya, O. G.: Hierarchical multi-label prediction of gene function, *Bioinformatics*, **22**, April 2006, 830–836, ISSN 1367-4803.
- [3] Bennett, P. N., Nguyen, N.: Refined experts: improving classification in large taxonomies, *SIGIR '09: Proceedings of the 32nd international ACM SIGIR con-*

ference on Research and development in information retrieval, ACM, New York, NY, USA, 2009, ISBN 978-1-60558-483-6.

- [4] Cai, L., Hofmann, T.: Hierarchical document categorization with support vector machines, *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, ACM, New York, NY, USA, 2004.
- [5] Ceci, M.: Hierarchical Text Categorization in a Transductive Setting, *ICDMW '08: Proceedings of the 2008 IEEE International Conference on Data Mining Workshops*, IEEE Computer Society, Washington, DC, USA, 2008, ISBN 978-0-7695-3503-6.
- [6] Ceci, M., Malerba, D.: Classifying web documents in a hierarchy of categories: a comprehensive study, *Journal of Intelligent Information Systems*, **28**(1), 2007, 37–78.
- [7] Chakrabarti, S., Dom, B., Agrawal, R., Raghavan, P.: Using Taxonomy, Discriminants, and Signatures for Navigating in Text Databases, *Proceedings of the 23rd VLDB Conference*, 1997.
- [8] Cheng, C., Tang, J., Fu, A., King, I.: Hierarchical Classification of Documents with Error Control, *PAKDD 2001 - Proceedings of 5th Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, 2001.
- [9] Clare, A., King, R.: Predicting gene function in *Saccharomyces cerevisiae*, *Bioinformatics*, **7**(19 (suppl. 2)), 2005, 42–49, ISSN 1361-8138.
- [10] D'Alessio, S., Murray, K., Schiaffino, R.: The Effect of Using Hierarchical Classifiers in Text Categorization, *Proceedings of the 6th International Conference on Recherche d'Information Assistée par Ordinateur (RIAO)*, 2000.
- [11] Deerwester, S., Dumais, S., Furnas, G., Landauer, T., Harshman, R.: Indexing by latent semantic analysis, *Journal of the American Society for Information Science*, **41**(6), 1990, 391–407.
- [12] Dekel, O., Keshet, J., Singer, Y.: Large Margin Hierarchical Classification, *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.
- [13] Dumais, S. T., Chen, H.: Hierarchical classification of Web content, *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval* (N. J. Belkin, P. Ingwersen, M.-K. Leong, Eds.), ACM Press, New York, US, Athens, GR, 2000.
- [14] Erik Wiener, Jan O. Pedersen, A. S. W.: A Neural Network Approach to Topic Spotting, *Proceedings of 4th Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas, US, 1995.
- [15] Esuli, A., Fagni, T., Sebastiani, F.: Boosting multi-label hierarchical text categorization, *Information Retrieval*, **11**(4), 2008, 287–313, ISSN 1386-4564.

- [16] Frommholz, I.: Categorizing Web Documents in Hierarchical Catalogues, *Proceedings of ECIR-01, 23rd European Colloquium on Information Retrieval Research*, 2001.
- [17] Golub, G. H., Kahan, W.: Calculating the singular values and pseudo-inverse of a matrix, *Journal of the Society for Industrial and Applied Mathematics*, **2**(2), 1965, 36–43, ISSN 205–224.
- [18] Ipeirotis, P., Gravano, L., Sahami, M.: Probe, count, and classify: categorizing hidden web databases, *SIGMOD Rec.*, **30**(2), 2001, 67–78.
- [19] Itskevitch, J.: *Automatic Hierarchical Email Classification Using Association Rules*, Ph.D. Thesis, Master’s thesis, Simon Fraser University, 2001.
- [20] Kiritchenko, S.: *Hierarchical text categorization and its application to bioinformatics*, Ph.D. Thesis, Univ. of Ottawa, Canada, Ottawa, Ont., Canada, Canada, 2006.
- [21] Koller, D., Sahami, M.: Hierarchically classifying documents using very few words, *Proceedings of ICML-97, 14th International Conference on Machine Learning* (D. H. Fisher, Ed.), Morgan Kaufmann Publishers, San Francisco, US, Nashville, US, 1997.
- [22] Kriegel, H., Kroger, P., Pryakhin, A., Schubert, M.: Using support vector machines for classifying large sets of multi-represented objects, *Proc. of the SIAM Int. Conference on Data Mining*, 2004.
- [23] Liu, H., Setiono, R.: Chi2: Feature selection and discretization of numeric attributes, *Proc. IEEE 7th International Conference on Tools with Artificial Intelligence*, 1995.
- [24] McCallum, A. K., Rosenfeld, R., Mitchell, T. M., Ng, A. Y.: Improving text classification by shrinkage in a hierarchy of classes, *Proceedings of ICML-98, 15th International Conference on Machine Learning* (J. W. Shavlik, Ed.), Morgan Kaufmann Publishers, San Francisco, US, Madison, US, 1998.
- [25] Ng, H. T., Goh, W. B., Low, K. L.: Feature selection, perceptron learning, and a usability case study for text categorization, *Proceedings of SIGIR-97, 20th ACM International Conference on Research and Development in Information Retrieval* (N. J. Belkin, A. D. Narasimhalu, P. Willett, Eds.), ACM Press, New York, US, Philadelphia, US, 1997.
- [26] Ruiz, M. E., Srinivasan, P.: Hierarchical Text Categorization Using Neural Networks, *Information Retrieval*, **5**(1), 2002, 87–118.
- [27] Sebastiani, F.: Machine Learning in Automated Text Categorization, *ACM Computing Surveys (CSUR)*, **34**(1), 2002, 1–55.

- [28] Silla, C. J., Freitas, A.: A survey of hierarchical classification across different application domains, *Journal of Data Mining and Knowledge Discovery*, **22**(1–2), 2010, 31–72.
- [29] Sun, A., Lim, E.: Hierarchical Text Classification and Evaluation, *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining*, IEEE Computer Society, Washington, DC, USA, 2001.
- [30] Sun, A., Lim, E., Ng, W., Srivastava, J.: Blocking reduction strategies in hierarchical text classification, *IEEE Transactions on Knowledge and Data Engineering*, **16**(10), 2004, 1305–1308.
- [31] Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y.: Support vector machine learning for interdependent and structured output spaces, *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, ACM, New York, NY, USA, 2004.
- [32] Valentini, G.: True path Rule Hierarchical Ensembles for Genome-Wide Gene Prediction, *IEEE/ACM Trans. on Comp. Biology and Bioinformatics*, **8**(3), 2011, 832–847.
- [33] Wang, K., Senqiang, K. W.: Hierarchical Classification of Real Life Documents, *Proc. of the 1st SIAM Int. Conference on Data Mining*, 2001.
- [34] Wang, K., Zhou, S., Liew, S.: Building Hierarchical Classifiers Using Class Proximity, *Proceedings of the 25th VLDB Conference*, Morgan Kaufmann Publishers, 1999.
- [35] Weigend, A. S., Wiener, E. D., Pedersen, J. O.: Exploiting Hierarchy in Text Categorization, *Information Retrieval*, **1**(3), 1999, 193–216.
- [36] Wibowo, W., Williams, H.: Strategies for Minimising Errors in Hierarchical Web Categorisation, *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, 2002.
- [37] Wibowo, W., Williams, H. E.: Simple and accurate feature selection for hierarchical categorisation, *DocEng '02: Proceedings of the 2002 ACM symposium on Document engineering*, ACM, New York, NY, USA, 2002.
- [38] Wu, F., Zhang, J., Honavar, V.: Learning Classifiers Using Hierarchically Structured Class Taxonomies, *Proc. of the Symp. on Abstraction, Reformulation, and Approximation*, 3607, Springer Verlag, 2005.

Appendix A. Estimate of the Joint Probability $p(X_k, \hat{X}_k)$

From the law of total probability, we can represent $p(X_k = i, \hat{X}_k = j) \equiv p(e_{ij}^{(k)})$ as follows:

$$p(e_{ij}^{(k)}) = \sum_{r,s} p(e_{rs}^{(k-1)}) \cdot p(e_{ij}^{(k)} | e_{rs}^{(k-1)})$$

Our goal is to derive an approximated model for $p(e_{ij}^{(k)})$. To differentiate between the actual probability and its approximation, the latter is denoted with $\omega_{ij}^{(k)}$.

For the sake of readability, we use “prime” to denote events or random variables that refer to a pipeline $\pi_k = \pi_{k-1} + c_k$, whereas plain text is used for π_{k-1} . Before deriving the estimation of the joint probability, let us recall that the following tautological implications hold:

$$\begin{aligned} X = 0 &\models X' = 0, & X' = 1 &\models X = 1 \\ \hat{X} = 0 &\models \hat{X}' = 0, & \hat{X}' = 1 &\models \hat{X} = 1 \end{aligned}$$

Estimate of $p(e'_{00})$

$$\triangleright p(e'_{00}) = p(e_{00}) \cdot p(e'_{00}|e_{00}) + p(e_{01}) \cdot p(e'_{00}|e_{01}) + p(e_{10}) \cdot p(e'_{00}|e_{10}) + p(e_{11}) \cdot p(e'_{00}|e_{11})$$

where:

$$\begin{aligned} p(e'_{00}|e_{00}) &= p(X' = 0, \hat{X}' = 0 | X = 0, \hat{X} = 0) \\ &= \underbrace{p(\hat{X}' = 0 | X' = 0, X = 0, \hat{X} = 0)}_{=1} \cdot \underbrace{p(X' = 0 | X = 0, \hat{X} = 0)}_{=1} = 1 \\ p(e'_{00}|e_{01}) &= p(X' = 0, \hat{X}' = 0 | X = 0, \hat{X} = 1) \\ &= \underbrace{p(\hat{X}' = 0 | X' = 0, X = 0, \hat{X} = 1)}_{\approx \gamma'_{00}} \cdot \underbrace{p(X' = 0 | X = 0, \hat{X} = 1)}_{=1} \approx \gamma'_{00} \\ p(e'_{00}|e_{10}) &= p(X' = 0, \hat{X}' = 0 | X = 1, \hat{X} = 0) \\ &= \underbrace{p(\hat{X}' = 0 | X' = 0, X = 1, \hat{X} = 0)}_{=1} \cdot \underbrace{p(X' = 0 | X = 1, \hat{X} = 0)}_{\approx \bar{f}'} \approx \bar{f}' \\ p(e'_{00}|e_{11}) &= p(X' = 0, \hat{X}' = 0 | X = 1, \hat{X} = 1) \\ &= \underbrace{p(\hat{X}' = 0 | X' = 0, X = 1, \hat{X} = 1)}_{\approx \gamma'_{00}} \cdot \underbrace{p(X' = 0 | X = 1, \hat{X} = 1)}_{\approx \bar{f}'} \approx \gamma'_{00} \cdot \bar{f}' \end{aligned}$$

Hence:

$$p(e'_{00}) \approx \omega'_{00} = \omega_{00} + \gamma'_{00} \cdot \omega_{01} + \bar{f}' \cdot \omega_{10} + \bar{f}' \cdot \omega_{11} \cdot \gamma'_{00}$$

Estimate of $p(e'_{01})$

$$\triangleright p(e'_{01}) = p(e_{00}) \cdot p(e'_{01}|e_{00}) + p(e_{01}) \cdot p(e'_{01}|e_{01}) + p(e_{10}) \cdot p(e'_{01}|e_{10}) + p(e_{11}) \cdot p(e'_{01}|e_{11})$$

where:

$$\begin{aligned} p(e'_{01}|e_{00}) &= p(X' = 0, \hat{X}' = 1 \mid X = 0, \hat{X} = 0) \\ &= \underbrace{p(\hat{X}' = 1 \mid X' = 0, X = 0, \hat{X} = 0)}_{=0} \cdot p(X' = 0 \mid X = 0, \hat{X} = 0) = 0 \\ p(e'_{01}|e_{01}) &= p(X' = 0, \hat{X}' = 1 \mid X = 0, \hat{X} = 1) \\ &= \underbrace{p(\hat{X}' = 1 \mid X' = 0, X = 0, \hat{X} = 1)}_{\approx \gamma'_{01}} \cdot \underbrace{p(X' = 0 \mid X = 0, \hat{X} = 1)}_{=1} \approx \gamma'_{01} \\ p(e'_{01}|e_{10}) &= p(X' = 0, \hat{X}' = 1 \mid X = 1, \hat{X} = 0) \\ &= \underbrace{p(\hat{X}' = 1 \mid X' = 0, X = 1, \hat{X} = 0)}_{=0} \cdot p(X' = 0 \mid X = 1, \hat{X} = 0) = 0 \\ p(e'_{01}|e_{11}) &= p(X' = 0, \hat{X}' = 1 \mid X = 1, \hat{X} = 1) \\ &= \underbrace{p(\hat{X}' = 1 \mid X' = 0, X = 1, \hat{X} = 1)}_{\approx \gamma'_{01}} \cdot \underbrace{p(X' = 0 \mid X = 1, \hat{X} = 1)}_{\approx \bar{f}} \approx \gamma'_{01} \cdot \bar{f}' \end{aligned}$$

Hence:

$$p(e'_{01}) \approx \omega'_{01} = 0 + \gamma'_{01} \cdot \omega_{01} + 0 + \bar{f}' \cdot \gamma'_{01} \cdot \omega_{11}$$

Estimate of $p(e'_{10})$

$$\triangleright p(e'_{10}) = p(e_{00}) \cdot p(e'_{10}|e_{00}) + p(e_{01}) \cdot p(e'_{10}|e_{01}) + p(e_{10}) \cdot p(e'_{10}|e_{10}) + p(e_{11}) \cdot p(e'_{10}|e_{11})$$

where:

$$\begin{aligned} p(e'_{10}|e_{00}) &= p(X' = 1, \hat{X}' = 0 \mid X = 0, \hat{X} = 0) \\ &= \underbrace{p(\hat{X}' = 0 \mid X' = 1, X = 0, \hat{X} = 0)}_{=0} \cdot p(X' = 1, \mid X = 0, \hat{X} = 0) = 0 \\ p(e'_{10}|e_{01}) &= p(X' = 1, \hat{X}' = 0 \mid X = 0, \hat{X} = 1) \\ &= \underbrace{p(\hat{X}' = 0 \mid X' = 1, X = 0, \hat{X} = 1)}_{=0} \cdot p(X' = 1 \mid X = 0, \hat{X} = 1) = 0 \end{aligned}$$

$$\begin{aligned}
p(e'_{10}|e_{10}) &= p(X' = 1, \widehat{X}' = 0 \mid X = 1, \widehat{X} = 0) \\
&= \underbrace{p(\widehat{X}' = 0 \mid X' = 1, X = 1, \widehat{X} = 0)}_{=1} \cdot \underbrace{p(X' = 1 \mid X = 1, \widehat{X} = 0)}_{\approx f'} \approx f' \\
p(e'_{10}|e_{11}) &= p(X' = 1, \widehat{X}' = 0 \mid X = 1, \widehat{X} = 1) \\
&= \underbrace{p(\widehat{X}' = 0 \mid X' = 1, X = 1, \widehat{X} = 1)}_{\equiv \gamma'_{10}} \cdot \underbrace{p(X' = 1 \mid X = 1, \widehat{X} = 1)}_{\approx \bar{f}'} \approx \gamma'_{10} \cdot \bar{f}'
\end{aligned}$$

Hence:

$$p(e'_{10}) \approx \omega'_{10} = 0 + 0 + f' \cdot \omega_{10} + f' \cdot \gamma'_{10} \cdot \omega_{11}$$

Estimate of $p(e'_{11})$

$$\triangleright p(e'_{11}) = p(e_{00}) \cdot p(e'_{11}|e_{00}) + p(e_{01}) \cdot p(e'_{11}|e_{01}) + p(e_{10}) \cdot p(e'_{11}|e_{10}) + p(e_{11}) \cdot p(e'_{11}|e_{11})$$

where:

$$\begin{aligned}
p(e'_{11}|e_{00}) &= p(X' = 1, \widehat{X}' = 1 \mid X = 0, \widehat{X} = 0) \\
&= \underbrace{p(\widehat{X}' = 1 \mid X' = 1, X = 0, \widehat{X} = 0)}_{=0} \cdot p(X' = 1 \mid X = 0, \widehat{X} = 0) = 0 \\
p(e'_{11}|e_{01}) &= p(X' = 1, \widehat{X}' = 1 \mid X = 0, \widehat{X} = 1) \\
&= \underbrace{p(\widehat{X}' = 1 \mid X' = 1, X = 0, \widehat{X} = 1)}_{=0} \cdot p(X' = 1 \mid X = 0, \widehat{X} = 1) = 0 \\
p(e'_{11}|e_{10}) &= p(X' = 1, \widehat{X}' = 1 \mid X = 1, \widehat{X} = 0) \\
&= \underbrace{p(\widehat{X}' = 1 \mid X' = 1, X = 1, \widehat{X} = 0)}_{=0} \cdot p(X' = 1 \mid X = 1, \widehat{X} = 0) = 0 \\
p(e'_{11}|e_{11}) &= p(X' = 1, \widehat{X}' = 1 \mid X = 1, \widehat{X} = 1) \\
&= \underbrace{p(\widehat{X}' = 1 \mid X' = 1, X = 1, \widehat{X} = 1)}_{\equiv \gamma'_{11}} \cdot \underbrace{p(X' = 1 \mid X = 1, \widehat{X} = 1)}_{\approx f'} \approx \gamma'_{11} \cdot f'
\end{aligned}$$

Hence:

$$p(e'_{11}) \approx \omega'_{11} = 0 + 0 + 0 + f' \cdot \gamma'_{11} \cdot \omega_{11}$$

Table 2 reports the patterns concerning co-occurring events that are certain or impossible to occur (all probabilities marked as 1 or 0 can be ascribed to one of these patterns). They are based on the following tautological implications:

$$\begin{aligned}
X = 0 &\models X' = 0 \\
\widehat{X} = 0 &\models \widehat{X}' = 0
\end{aligned}$$

Table 2: Patterns for certain or impossible events

Certain Events	Impossible Events
$p(X' = 0 \mid X = 0, \dots) = 1$	$p(\dots \mid X' = 1, X = 0, \dots) = 0$
$p(\widehat{X}' = 0 \mid \dots, \widehat{X} = 0) = 1$	$p(X' = 1 \mid X = 0, \dots) = 0$
	$p(\widehat{X}' = 1 \mid \dots, \widehat{X} = 0) = 0$

Table 3 reports the patterns concerning the approximations made while deriving the joint probability $p(X_k, \widehat{X}_k)$, the underlying hypothesis being that a high correlation holds between the involved classifiers and the corresponding oracles. In particular, in presence of co-occurring events such as $\langle X = 1, \widehat{X} = 1 \rangle$, this assumption permits to disregard $X = 1$ or $\widehat{X} = 1$.

Table 3: Approximation Patterns

Pattern	Approximation
$p(\widehat{X}' = j \mid X' = i, X = 1, \widehat{X} = 1)$	$p(\widehat{X}' = j \mid X' = i, \widehat{X} = 1) = \gamma'_{ij}, \quad i, j = 0, 1$
$p(X' = 1 \mid X = 1, \widehat{X} = 1)$	$p(X' = 1 \mid X = 1) = f'$
$p(X' = 0 \mid X = 1, \widehat{X} = 1)$	$p(X' = 0 \mid X = 1) = \bar{f}'$
$p(\widehat{X}' = j \mid X' = 0, X = 0, \widehat{X} = 1)$	$p(\widehat{X}' = j \mid X' = 0, \widehat{X} = 1) = \gamma'_{0j}, \quad j = 0, 1$
$p(X' = 1 \mid X = 1, \widehat{X} = 0)$	$p(X' = 1 \mid X = 1) = f'$
$p(X' = 0 \mid X = 1, \widehat{X} = 0)$	$p(X' = 0 \mid X = 1) = \bar{f}'$

Other approximations have been made by exploiting also the total probability law. As an example, let us assume that we want to find an approximation for:

$$p(\widehat{X}' = 0 \mid X' = 0, X = 0, \widehat{X} = 1)$$

We know that, by hypothesis:

$$\gamma'_{00} = p(\widehat{X}' = 0 \mid X' = 0, \widehat{X} = 1)$$

Hence, we can write:

$$\gamma'_{00} = p(\widehat{X}' = 0, X = 0 \mid X' = 0, \widehat{X} = 1) + p(\widehat{X}' = 0, X = 1 \mid X' = 0, \widehat{X} = 1)$$

With $\alpha \triangleq p(X = 1 | X' = 0, \hat{X} = 1)$ and recalling that $X = 1$ and $\hat{X} = 1$ are highly correlated by hypothesis, we can write:

$$\gamma'_{00} = p(\hat{X}' = 0 | X' = 0, X = 0, \hat{X} = 1) \cdot (1 - \alpha) + \underbrace{p(\hat{X}' = 0 | X' = 0, X = 1, \hat{X} = 1)}_{\approx \gamma'_{00}} \cdot \alpha$$

Which yields:

$$p(\hat{X}' = 0 | X' = 0, X = 0, \hat{X} = 1) \approx \gamma'_{00}$$

Appendix B. Deriving the Ψ Homomorphism

Given a taxonomy $\mathcal{T} = \langle \mathcal{C}, \leq \rangle$, it is well known that the closure of \mathcal{C} under the Kleene star (i.e., \mathcal{C}^*) yields a monoid, with:

1. Closure (wrt the operator “+”):
 $\forall \pi_1, \pi_2 \in \mathcal{C}^* : \pi_1 + \pi_2 \in \mathcal{C}^*$
2. Associativity (wrt the operator “+”):
 $\forall \pi_1, \pi_2, \pi_3 \in \mathcal{C}^* : (\pi_1 + \pi_2) + \pi_3 = \pi_1 + (\pi_2 + \pi_3)$
3. Neutral element (empty string λ):
 $\forall \pi \in \mathcal{C}^* : \pi + \lambda = \lambda + \pi = \pi$

In the event that Ψ is a homomorphism, also the set of normalized confusion matrices $\mathcal{M} \equiv [0, 1]^4$ is a monoid, as a homomorphism is expected to preserve the structure while mapping \mathcal{C}^* to \mathcal{M} . Let us verify that Ψ is a homomorphism by checking whether the space \mathcal{M} is a monoid, with “+” \rightarrow “ \oplus ” and $\lambda \rightarrow \mu$:

1. Closure (wrt the operator “ \oplus ”):
 $\forall a, b \in \mathcal{M} : a \oplus b \in \mathcal{M}$
2. Associativity (wrt the operator “ \oplus ”):
 $\forall a, b, c \in \mathcal{M} : (a \oplus b) \oplus c = a \oplus (b \oplus c)$
3. Neutral element (neutral classifier μ):
 $\forall a \in \mathcal{C}^* : a \oplus \mu = \mu \oplus a = a$

Proof.

1. Closure under “ \oplus ”: $\alpha, \beta \in \mathcal{M} \Rightarrow \alpha \oplus \beta \in \mathcal{M}$

$$\alpha \oplus \beta = \begin{bmatrix} \alpha_{00} + \alpha_{01} \cdot \beta_{00} & \alpha_{01} \cdot \beta_{01} \\ \alpha_{10} + \alpha_{11} \cdot \beta_{10} & \alpha_{11} \cdot \beta_{11} \end{bmatrix}$$

where

$$\begin{aligned} 0 \leq (\alpha \oplus \beta)_{00} \leq \alpha_{00} + \alpha_{01} &= 1, & 0 \leq (\alpha \oplus \beta)_{01} \leq \alpha_{01} &\leq 1 \\ 0 \leq (\alpha \oplus \beta)_{10} \leq \alpha_{10} + \alpha_{11} &= 1, & 0 \leq (\alpha \oplus \beta)_{11} \leq \alpha_{11} &\leq 1 \end{aligned}$$

Moreover:

$$\begin{aligned} (\alpha \oplus \beta)_{00} + (\alpha \oplus \beta)_{01} &= (\alpha_{00} + \alpha_{01} \cdot \beta_{00}) + \alpha_{01} \cdot \beta_{01} = \alpha_{00} + \alpha_{01} \cdot (\beta_{00} + \beta_{01}) = 1 \\ (\alpha \oplus \beta)_{10} + (\alpha \oplus \beta)_{11} &= (\alpha_{10} + \alpha_{11} \cdot \beta_{10}) + \alpha_{11} \cdot \beta_{11} = \alpha_{10} + \alpha_{11} \cdot (\beta_{10} + \beta_{11}) = 1 \end{aligned}$$

2. Associativity under “ \oplus ”: $(\alpha \oplus \beta) \oplus \gamma = \alpha \oplus (\beta \oplus \gamma)$

$$\begin{aligned}
(\alpha \oplus \beta) \oplus \gamma &= \begin{bmatrix} \alpha_{00} + \alpha_{01} \cdot \beta_{00} & \alpha_{01} \cdot \beta_{01} \\ \alpha_{10} + \alpha_{11} \cdot \beta_{10} & \alpha_{11} \cdot \beta_{11} \end{bmatrix} \oplus \begin{bmatrix} \gamma_{00} & \gamma_{01} \\ \gamma_{10} & \gamma_{11} \end{bmatrix} \\
&= \begin{bmatrix} \alpha_{00} + \alpha_{01} \cdot \beta_{00} & 0 \\ \alpha_{10} + \alpha_{11} \cdot \beta_{10} & 0 \end{bmatrix} + \begin{bmatrix} \alpha_{01} \cdot \beta_{01} & 0 \\ 0 & \alpha_{11} \cdot \beta_{11} \end{bmatrix} \cdot \begin{bmatrix} \gamma_{00} & \gamma_{01} \\ \gamma_{10} & \gamma_{11} \end{bmatrix} \\
&= \begin{bmatrix} \alpha_{00} + \alpha_{01} \cdot \beta_{00} + \alpha_{01} \cdot \beta_{01} \cdot \gamma_{00} & \alpha_{01} \cdot \beta_{01} \cdot \gamma_{01} \\ \alpha_{10} + \alpha_{11} \cdot \beta_{10} + \alpha_{11} \cdot \beta_{11} \cdot \gamma_{10} & \alpha_{11} \cdot \beta_{11} \cdot \gamma_{11} \end{bmatrix} \\
\alpha \oplus (\beta \oplus \gamma) &= \begin{bmatrix} \alpha_{00} & \alpha_{01} \\ \alpha_{10} & \alpha_{11} \end{bmatrix} \oplus \begin{bmatrix} \beta_{00} + \beta_{01} \cdot \gamma_{00} & \beta_{01} \cdot \gamma_{01} \\ \beta_{10} + \beta_{11} \cdot \gamma_{10} & \beta_{11} \cdot \gamma_{11} \end{bmatrix} \\
&= \begin{bmatrix} \alpha_{00} & 0 \\ \alpha_{10} & 0 \end{bmatrix} + \begin{bmatrix} \alpha_{01} & 0 \\ 0 & \alpha_{11} \end{bmatrix} \cdot \begin{bmatrix} \beta_{00} + \beta_{01} \cdot \gamma_{00} & \beta_{01} \cdot \gamma_{01} \\ \beta_{10} + \beta_{11} \cdot \gamma_{10} & \beta_{11} \cdot \gamma_{11} \end{bmatrix} \\
&= \begin{bmatrix} \alpha_{00} + \alpha_{01} \cdot \beta_{00} + \alpha_{01} \cdot \beta_{01} \cdot \gamma_{00} & \alpha_{01} \cdot \beta_{01} \cdot \gamma_{01} \\ \alpha_{10} + \alpha_{11} \cdot \beta_{10} + \alpha_{11} \cdot \beta_{11} \cdot \gamma_{10} & \alpha_{11} \cdot \beta_{11} \cdot \gamma_{11} \end{bmatrix}
\end{aligned}$$

3. Neutral element μ : $\alpha \in \mathcal{M} \Rightarrow \alpha \oplus \mu = \mu \oplus \alpha \equiv \alpha$, with $\mu = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$

The neutral element μ corresponds to a classifier that accepts and passes down all its input data (i.e., FP and TP). It is easy to verify that this property holds for the choice made about μ :

$$\alpha \oplus \mu = \begin{bmatrix} \alpha_{00} & \alpha_{01} \\ \alpha_{10} & \alpha_{11} \end{bmatrix} \oplus \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha_{00} & 0 \\ \alpha_{10} & 0 \end{bmatrix} + \begin{bmatrix} \alpha_{01} & 0 \\ 0 & \alpha_{11} \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \equiv \alpha$$

$$\mu \oplus \alpha = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \oplus \begin{bmatrix} \alpha_{00} & \alpha_{01} \\ \alpha_{10} & \alpha_{11} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \alpha_{00} & \alpha_{01} \\ \alpha_{10} & \alpha_{11} \end{bmatrix} \equiv \alpha$$